INPUT

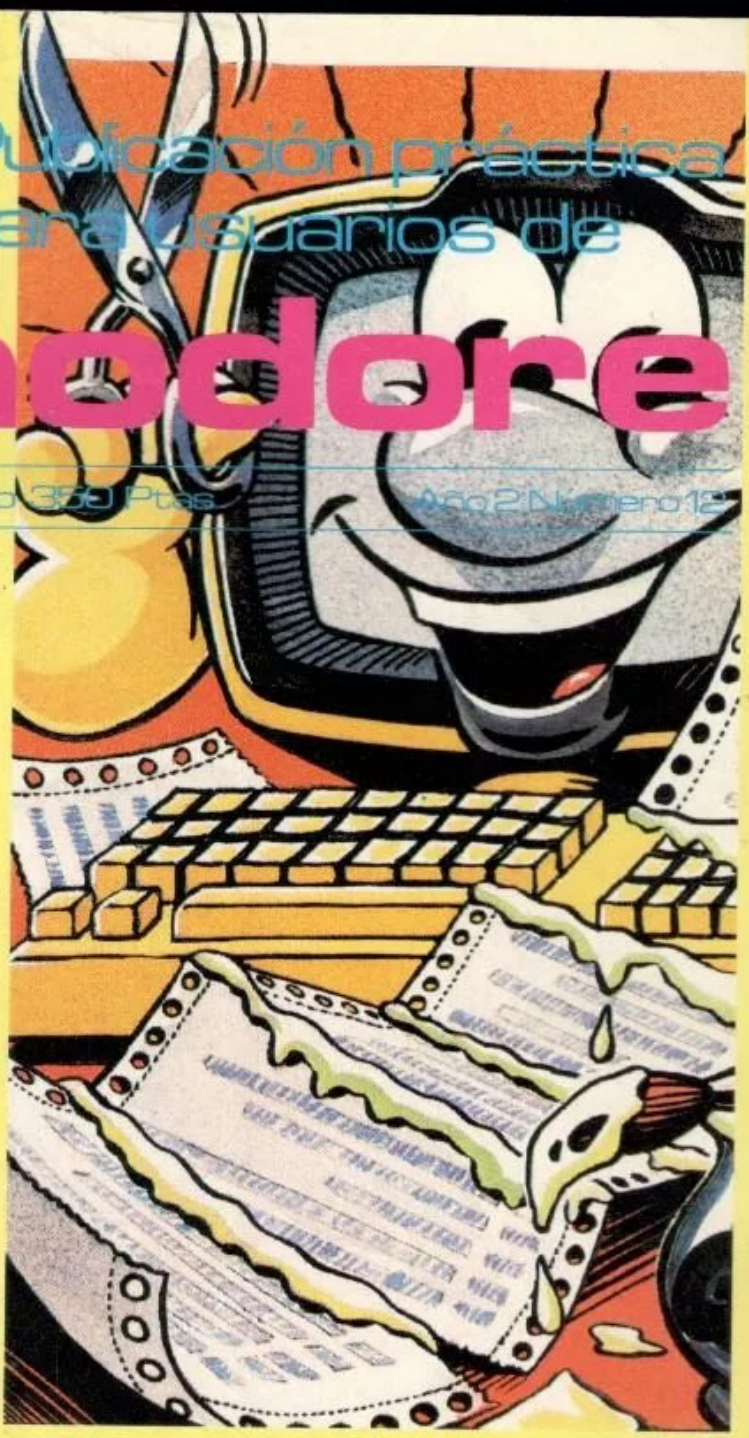
Publicación práctica para usuarios de

commodore

Revista mensual 1986 Precio 350 Ptas. Año 2 Número 12

ADAPTADOR PARA DISCO

SOLIDOS DE REVOLUCION



TALIZADOR EL P.C.

Ha encontrado su perdición

Hasta ahora, en ERBE siempre habíamos preferido sacar juegos de acción a los de simulación. Ya sabes... parece más divertido un GREEN BERET, un WAY OF THE TIGER o un BAT-MAN que uno de esos complicados simuladores, en los que antes de jugarlos tienes que esperar a que te crezcan 7 dedos en cada mano para poder manejar los controles, y armarte de valor para leer esa guía telefónica bautizada como "instrucciones".

Lo que son las cosas... Una mañana llegaron a las oficinas de ERBE los señores de U.S. GOLD, y con voz trémula por la emoción (los ingleses siempre tienen la voz trémula por la emoción) dijeron:

—Misco fla fla tirurero flandis. (Aquí os traemos un juego que hará historia.)

—Menos lobos, Caperucita, y decirnos de qué se trata—contestamos nosotros.

—Chunguili biri ***LEADER BOARD*** *manguis turrutis. (Ni más ni menos que* ***LEADER BOARD,*** *el mejor simulador de golf que hayáis visto.)*

Fuertes mareos y náuseas nos invadieron. No te fastidia... un simulador y, además, de golf. ¡Vaya muermo!

De repente, alguno de nosotros (todavía no sabemos quién fue) dijo:

—Vamos a verlo. A lo mejor se lo podemos vender a los padres para que duerman a sus hijos, o les amenacen con ponérselo si no terminan los deberes... y ésa fue la perdición.

Tan pronto como se cargó **LEADER BOARD** empezaron a desmoronarse nuestras más firmes creencias. El golf no era un muermo, ni siquiera aburrido, empezaba a gustarnos, algunos incluso se apasionaban... y es que **LEADER BOARD** era **fantástico.**

Descubrimos que podíamos elegir entre 78 hoyos distintos en 4 circuitos diferentes, que **LEADER BOARD** nos permitía controlar la fuerza y el efecto de cada golpe que diéramos, con cada uno de los 14 palos distintos entre los que podíamos elegir, que con LEADER BOARD podíamos optar a tres niveles de dificultad, en el último de los cuales hasta el viento y la inclinación del "green" debían tenerse en cuenta, y que su manejo era sencillísimo..., y que los señores de U.S. GOLD tenían razón: **LEADER BOARD es un juego que hará historia...**

Aunque sólo sea porque a ERBE le va a buscar la perdición, puesto que desde ese día nuestros eminentes directivos, nuestros cerebros pensantes, están delante de la pantalla con los ojos inyectados en sangre, jurando y perjurando que el próximo hoyo lo harán bajo par.

¡¡¡SI SEVERIANO LOS VIERA...!!!

En serio, **LEADER BOARD** es el mejor simulador de golf que hayáis visto.

¡¡¡ PALABRA DE ERBE !!!

!!! PALABRA DE U.S. GOLD !!!

DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA: ERBE SOFTWARE. C/. STA. ENGRACIA, 17
28010 MADRID. TEL. (91) 447 34 10 - DELEGACION BARCELONA, AVDA. MISTRAL, N.º 10 - TEL. (93) 432 07 31

AÑO 1 NUMERO 12

DIRECTOR:
Alejandro Diges
COORDINADOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Luis R. Palencia, Christophe Païs, Francisco Tórtola, Benito Román, Esther de la Cal.
PORTADA:
Foto, Nieves; Maquillaje y peluquería, Grupo Cinco; Modelo, Susana Alguacil.
INPUT Commodore es una publicación juvenil de EDICIONES FORUM
GERENTE DIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: Grupo Jota
Madrid: c/ General Varela, 35
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.º
Teléf. 250 23 99
FOTOMECANICA: Ochoa, S. A.
COMPOSICION: EFCA, S. A.
IMPRESION: Sirven Grafic
C/ Gran Vía, 754-756. 08013 Barcelona
Depósito legal: M. 27.884-1985
SUSCRIPCIONES: EDISA,
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12
REDACCION:
Paseo de la Castellana, 93, 14.ª
28046 Madrid. Teléf. 456 54 13
DISTRIBUIDORA
R.B.A. PROMOTORA DE EDICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.
El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.
Se ha solicitado el control OJD

INPUT Commodore es independiente y no está vinculada a Commodore Business Machines o sus distribuidores.
INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío.
Copyright ilustraciones del fondo gráfico de Marshall Cavendish, págs. 22, 23, 24, 26, 27, 31, 32, 33, 34, 36, 37, 40, 41, 42, 43, 44, 47.

INPUT commodore

SUMARIO

EDITORIAL 4

PROGRAMACION
TODO SOBRE READ Y DATA 6
SOLIDOS DE REVOLUCION 10
PREDICIENDO LO IMPREDECIBLE 39
EL CODIGO ASCII 50

REVISTA DE HARDWARE
1541 FANTASTICA 18
DIGITALIZADOR DE IMAGEN 48

CODIGO MAQUINA
ADAPTADOR DE DISCO 20
APROVECHA LAS INTERRUPCIONES 44
ARITMETICA HEXADECIMAL 14

REVISTA DE SOFTWARE 3
GAMES MAKER 56
PROGRAMAS 58

EL ZOCO 64

PROGRAMACION DE JUEGOS (COLECCIONABLE) 31
DECODIFICACION DE TEXTOS
UTILIZA TU COMPRESOR

EL PRIMER AÑO

Con éste son ya doce números en la calle; mejor dicho: trece. El especial de verano fue añadido como consecuencia de vuestra inestimable aceptación. Igualmente tuvimos en cuenta el éxito del cuadernillo de programas que incluíamos en el ejemplar del pasado diciembre. De ahí salió un número diferente que potencia la fiel continuidad de la cita mensual.

El balance de estos doce meses nos hace darnos cuenta de vuestro alto índice de participación en las actividades a las que os hemos convocado, desde concursos, votaciones..., sin olvidarnos de la abrumadora respuesta a la encuesta.

Hemos publicado unas cuantas colaboraciones de lectores cuyo excelente nivel las hizo pasar desapercibidas (para bien) entre los artículos desarrollados por autores con experiencia. Sin embargo, la cantidad de cintas recibidas con programas vuestros, no siendo pocas, distan algo de nuestras expectativas. Os animamos a enviarlos, un buen programa no puede dormitar en un cajón.

Por otro lado, durante tan corto período, hay que ver cómo ha evolucionado este mundo nuestro, con la aparición de nuevas marcas y modelos de ordenador más competitivos y, sobre todo, lo que es casi increíble ha sido la evolución de los precios a la baja.

También es cierto que se han malogrado algunos otros sistemas, en los que se habían puesto grandes esperanzas por su aparente competitividad y todos conocemos cuáles son.

Este número vuelve a seguir la línea de los once anteriores, que se caracteriza por su intención didáctica frente a la opción de publicar fundamentalmente listados para copiar, sin rehuír suministrar algunos de vez en cuando.

Desde este editorial os volvemos a instar para que continuéis enviando vuestras colaboraciones y programas. Estamos seguros de que habéis preparado trabajos sensacionales durante el estío.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** Alberto Alcocer, 46 - 4.º B. 28016 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido _____ **Segundo título elegido** _____

Tercer título elegido _____ **Programa que te gustaría conseguir** _____

Qué ordenador tienes _____ **Nombre** _____

1.er Apellido _____ **2.º Apellido** _____

Fecha de nacimiento _____ **Teléfono** _____

Dirección _____ **Localidad** _____

Provincia _____

LO MEJOR PARA COMMODORE

Spindizzy

EL RETO A TU DESTREZA EN TRES DIMENSIONES

— Con SPINDIZZY te trasportaras a un enigmático mundo de fantasía.
— Sus casi 400 pantallas se convierten en fuente inagotable de diversión.
— Los mejores gráficos tridimensionales que un AMSTRAD puede generar en un juego de este tipo.
— Tu mismo puedes alterar el trazado de las pantallas, situándote cada vez ante un nuevo desafío: rampas, ascensores, trampas, etc.

TAMBIEN DISPONIBLE PARA AMSTRAD Y SPECTRUM

Electric Dreams SOFTWARE

EN TIENDAS ESPECIALIZADAS Y GRANDES ALMACENES, O DIRECTAMENTE POR CORREO O TELEFONO A:

PROEIN, S.A.

Velázquez, 10 - 28001 Madrid Tels. (91) 276 22 08/09

TODO SOBRE READ Y DATA

- COMO FUNCIONAN READ Y DATA
- LA SENTENCIA RESTORE
- COMO CONSTRUIR GRAFICOS SENCILLOS A PARTIR DE SENTENCIAS DATA

Haciendo que el ordenador lea por medio de la sentencia READ un gran volumen de datos te puedes ahorrar tener que teclear programas largos. Puedes usar esta misma técnica para todo, desde gráficos hasta índices sencillos.

Una de las características que hace que un ordenador sea tan versátil es el uso de variables. Y casi siempre la asignación de valores a las variables es algo extremadamente sencillo, no tienes más que escribir, por ejemplo, LET X = 5. Pero hay ocasiones en que la cantidad de información que se quiere utilizar es tan grande que llega a desbordarte. Aquí es donde encuentran su gran utilidad las sentencias DATA y sus acompañantes READ y RESTORE.

La palabra DATA tiene aquí un significado muy específico. Se suele utilizar el término «datos» para designar una gran variedad de cosas. Por ejemplo, un programa, una rutina en código máquina y una matriz almacenada en cinta o *diskette*, a todo esto se le suele llamar a veces «datos». Pero en este artículo al hablar de datos, nos estamos refiriendo a las sentencias DATA y a los elementos que contienen.

El primer método que se suele aprender para asignar valores a una variable, ya se trate de una variable numérica o de una cadena de caracteres, suele ser por medio de una sentencia INPUT. Pero ésta sólo resulta de utilidad cuando es el propio usuario el que suministra la información al ordenador cada vez que se ejecuta un programa.

Muchas veces sin embargo, hay valores fijos que no tienen que ser introducidos ni modificados por el operador, por lo que se pueden incorporar permanentemente en el programa. Este es naturalmente el caso típico de aplicación de la sentencia LET. Pero cuando el volumen de información es muy grande, el uso de grandes masas de sentencias LET es muy tedioso de programar y además la ejecución del programa resulta bastante lenta.

Aquí tienes un ejemplo en el que se podría utilizar una lista fija de encabezamientos en un programa para imprimir un nuevo documento:

Teclea para C-64 y Vic-20

```
10 LET A$="DIA"
20 LET B$="SEMANA"
30 LET C$="MES"
40 LET D$="A#O"
50 PRINT A$,B$,C$,D$
```

Pero también podrías utilizar sentencias DATA para conseguir el mismo resultado final:

```
10 READ A$,B$,C$,D$
20 PRINT A$.,B$,C$,D$
100 DATA DIA,SEMANA,MES,A#O
```

Si ahora resultara que quisieras utilizar no cuatro, sino cincuenta encabezamientos diferentes, con el primer programa se requerirían 46 sentencias adicionales, mientras que con el segundo sólo harían falta una o dos más, dependiendo del ordenador y del tamaño de los encabezamientos.

COMO FUNCIONAN LAS SENTENCIAS «READ» Y «DATA»

¿Cómo funcionan realmente las sentencias READ y DATA? Cuando el ordenador se encuentra con una ins-

trucción READ, explora todo el programa hasta que encuentra la primera sentencia DATA. A continuación asigna el valor del primer dato de dicha sentencia a la variable que figura en la sentencia READ.

Así, en el programa anterior la cadena de caracteres «DIA» queda asignada a la variable A$, a continuación se asigna «SEMANA» a la variable B$, etc.

Las sentencias DATA aparecen normalmente al final de los programas para «no estorbar» mientras el resto del programa se encuentra en fase de escritura y depuración. Pero realmente pueden ir situadas en cualquier parte. El ordenador simplemente ignora todas las sentencias DATA a menos que una sentencia READ le ordene hacer lo contrario. El siguiente programa funcionaría perfectamente:

Teclea para C-64 y Vic-20

```
10 DATA ALEMANIA
20 READ A$,B$
30 DATA FRANCIA,ITALIA,ESPA#A
40 READ C$,D$
```

Sin embargo es evidente que el programa resultará mucho más fácil de leer si todas las sentencias DATA se ponen agrupadas en alguna parte. Además hay una regla que respetar siempre: los datos de las sentencias DATA han de aparecer en el orden en que el ordenador debe leerlos.

El número de elementos que puede contener una sola lista de datos depende de la máxima longitud de línea del ordenador. En cuanto una línea esté llena se empieza una nueva; el ordenador sigue tomándolas en orden.

UNA LISTA SENCILLA

Aquí tienes un programa que utiliza un bucle para leer los datos con las sentencias READ y DATA. Es un programa muy sencillo de lista de teléfonos. Tú introduces el nombre de la persona y el ordenador te presenta su número de teléfono.

Teclea para C-64 y Vic-20

```
5 PRINT"[SHIFT+CLR/HOME]"
10 PRINT TAB(10)"DIRECTORIO
   TELEFONICO"
20 INPUT"[CRSR abajo]ESCRIBE
   EL NOMBRE";R$
25 PRINT
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN PRINT"EL
   NUMERO DE ";N$;" ES";T$:
   END

60 IF N$="FIN" THEN PRINT R$
   ;" NO ESTA EN LA LISTA"
70 NEXT J
500 DATA PEPE,425 12 34,LUIS
    ,876 23 45,ARTURO, 567
    34 56,RAUL,784 34 12,FIN
    ,FIN
```

Teclea el programa, pero ten la precaución de utilizar en la línea DATA los nombres y números de teléfonos de tus amigos. Puedes poner todos los datos que quieras. Lo único que tienes que cuidar al final es de ajustar correctamente el contador de bucle de la línea 30.

Observa que la lista de datos se termina con FIN,FIN. De esta forma la línea 60 puede comprobar si se ha alcanzado el final de la línea. Si el programa ha podido leer hasta aquí, significa que no ha encontrado el nombre, por lo que visualiza un mensaje para decírtelo. Hay que introducir FIN dos veces, debido a que la línea 40 lee los datos de dos en dos, con lo que de no ser así obtendrás un mensaje de error de falta de datos.

Fíjate también en que para leer un número de teléfono se considera a éste como una cadena de caracteres. Se ha hecho así para respetar un signo de puntuación, el guión, que evidentemente no puede figurar como un número. Las cadenas de caracteres que figuren dentro de una sentencia DATA pueden contener espacios, pero no pueden contener comas. Como los diferentes datos están separados por comas, el ordenador consideraría la coma como el final de uno de los datos. Si te vieras obligado a introducir en una sentencia DATA una cadena de caracteres que contenga una coma, tienes que encerrar toda esa cadena entre comillas. Sería necesario hacer esto en una sentencia DATA como la que figura en el ejemplo siguiente:

Teclea para C-64 y Vic-20

```
10 READ N$,A$,B$,C$
20 DATA LUIS PELAEZ,"C/PINTO
   , 25", MURCIA
```

La primera línea de la dirección ha de ir entre comillas para poder incluir la coma que figura a continuación del número 23.

USO DE «RESTORE»

Con los métodos que llevamos explicados hasta este momento, un programa es capaz de leer mediante una sentencia READ la correspondiente lista de datos de la sentencia DATA, pero sólo lo hará una vez a menos que se vuelva a ejecutar de nuevo el programa. Si quieres que se vuelvan a leer los datos, debes incorporar en tu programa una sentencia RESTORE.

Supongamos por ejemplo que quieres mirar los números de teléfono de algunos de tus amigos. La forma inmediata de hacerlo es poner en el programa una rutina de «Otra Vez». Pon primero END o STOP en la línea 50 junto con GOTO 80, y a continuación añade las siguientes líneas de programa:

Teclea para C-64 y Vic-20

```
80 PRINT TAB(4)"QUIERES OTRO
   NUMERO (S/N)?"
90 GET K$:IF K$="" THEN 90
100 IF K$="S" THEN GOTO 5
110 END
```

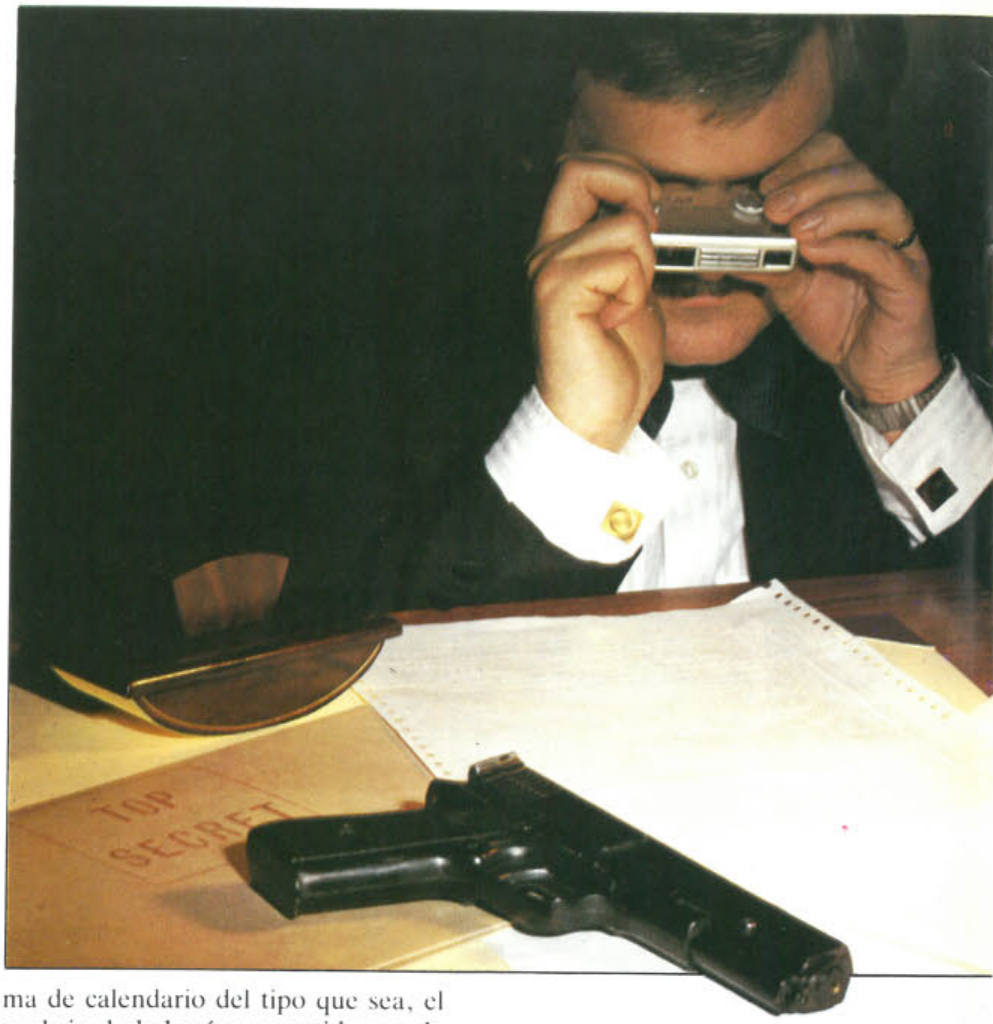
¿Qué es lo que sucede cuando intentas ejecutar el programa? La primera vez el programa funciona perfectamente. Pero cuando pulsas una tecla para intentar ejecutarlo por segunda vez, te falla porque le faltan datos. Ello se debe a que durante la primera ejecución llegó hasta el final de la lista de datos. Afortunadamente hay una solución muy sencilla para este problema. Teclea:

```
15 RESTORE
```

Esta vez el programa funciona bien todas las veces. Ello se debe a que la instrucción RESTORE hace que el ordenador vuelva hasta el principio de la lista de datos.

Es una buena idea poner un RESTORE cerca del principio del programa durante la fase de desarrollo. Esto te evita tener que recorrer todas las listas de datos en las pruebas de ejecución. Si crees que más adelante te va a convenir suprimir dicha línea de RESTORE, pon una sentencia REM con un mensaje para que te lo recuerde.

Por medio de RESTORE, puedes volver a utilizar una lista de datos tantas veces como sea necesario. Esto resulta particularmente útil en programas que tienen etiquetas, encabezamientos, tablas, etc, que tienen que ser repetidas en momentos diferentes. Por ejemplo, puedes poner los meses del año en una lista de datos dentro de una sentencia DATA en un programa de calendario del tipo que sea, el cual sin duda hará un repetido uso de dichas tablas.



La instrucción RESTORE es especialmente útil en los casos en los que deseas seguir leyendo con READ una lista para buscar un determinado elemento, como es el caso del programa de lista telefónica que hemos visto anteriormente.

USO DE LA SENTENCIA «DATA»

Las listas de datos que contienen las sentencias DATA son extremadamente útiles en toda clase de programas. Puede que ya hayas hecho uso de ellas en los programas de gráficos definidos por el usuario y en el dibujo del laberinto. También hemos visto ya cómo puede leer el ordenador los parámetros de la música y otros efectos sonoros.

En los juegos de aventuras, se suele utilizar con frecuencia gran cantidad de líneas de datos en las que se pone todo el texto necesario. Los juegos de marcianitos escritos en BASIC, también usan con frecuencia las sentencias DATA para definir personajes y otros elementos del juego.

Los programadores más expertos utilizan las sentencias DATA para hacer programas en código máquina o en lenguaje ensamblador. Tales programas pueden consistir en un corto bucle que contiene comandos POKE los cuales leen (mediante READ) los datos de las sentencias DATA.

En una palabra, todo programa que contenga texto ordinario, números o funciones, puede hacer uso con gran provecho de las listas de datos. Con un uso cuidadoso estas listas pueden convertirse en una poderosa herramienta de programación.

USO DE LOS «DATA» EN LOS GRAFICOS

Las sentencias DATA son muy útiles en los programas de gráficos para fijar las coordenadas que hay que dibujar y para llamar a las rutinas gráficas de la ROM. El uso de los DATA de esta manera resulta ideal para las formas irregulares como las que figuran en la siguiente página, ya que se necesitarían muchas líneas de programa si sólo se emplearan instrucciones individuales de impresión o dibujo.

No obstante existen casos en que una sentencia DATA no es de gran utilidad. Así ocurre cuando estás dibujando formas muy regulares en las que las coordenadas o el contorno gráfico pueden calcularse fácilmente.

Las sentencias DATA del programa que veremos a continuación, han sido divididas deliberadamente en varias líneas. Se ha hecho así porque la posterior modificación de los datos puede ser difícil a menos que se tengan unas divisiones muy claras; imagínate lo que sería volver sobre este programa dentro de unos meses e intentar localizar y modificar unos cuantos datos de cada una de las variables. Naturalmente, puedes seguir detenidamente la pista de cada una de las instrucciones READ, pero hacer esto en un programa largo es muy tedioso. Por ello te recomendamos que fragmentes cada grupo de sentencias DATA en varias líneas y que además (suponiendo que tengas memoria disponible) utilices abundantemente las sentencias REM para que las cosas te resulten más claras.

El siguiente programa utiliza una rutina READ ... DATA para dibujar una pequeña casa de campo a partir de algunos de los gráficos de ROM que tiene el **Commodore 64**.

Teclea para Commodore-64

```
10 PRINT"[SHIFT+CLR/HOME]"
20 POKE 53280,3:POKE
   53281,5
30 FOR A=1 TO 7
40 FOR B=1 TO 14
50 READ X
60 POKE 1024+291+(A*40)+B,X
65 POKE 55296+291+(A*40)
   +B,7
70 NEXT B
75 NEXT A
500 DATA 32,32,32,32,32,32,
    32,32,32,32,108,160,123,
    32
505 DATA 32,32,233,160,223,
    32,32,32,32,118,160,160,
    160,116
510 DATA 32,233,160,160,160,
    223,32,92,32,160,160,
    160,160,160
520 DATA 32,102,102,102,102,
    102,160,160,223,108,160,
    102,160,120
530 DATA 32,102,91,102,172,
    102,102,91,102,32,32,102
    ,32,32
540 DATA 32,102,102,102,160,
    102,102,102,102,32,32,
    102,32,32
550 DATA 99,99,99,99,99,99,
    99,99,99,99,99,99
    ,99,99
```

En el programa anterior se utilizan siete líneas de sentencias DATA las cuales corresponden a las siete filas de la pantalla que forman parte del bucle FOR ... NEXT que comienza en la línea 30. Los catorce elementos que figuran dentro de cada DATA corresponden a las catorce columnas especificadas en el bucle FOR ... NEXT que se origina en la línea 40.

Cuando el valor de X, que es un código de pantalla (consulta tu manual), ha sido leído con arreglo a la instrucción de la línea 50, se pokea en la memoria de pantalla tal como ordena la línea 60. También hay que poner en la memoria de color un POKE correspondiente al carácter que se desea mostrar: esto es lo que se hace en la línea 65. En estas dos líneas el número 291 sirve para iniciar el gráfico en un punto cercano al centro de la pantalla, si bien la posición real viene dictada por los valores de las variables A y B. Teclea ahora las siguientes líneas:

Teclea para Commodore-64

```
220 FOR K=1 TO 2000
230 NEXT K
240 PRINT"[SHIFT+CLR/HOME]
    [CTRL+3] PRESIONA CUAL
    QUIER TECLA PARA SEGUIR"
250 GET K$:IF K$=""THEN GOTO
    250
270 GOTO 10
```

Ejecuta el programa y contesta al mensaje de pantalla después de transcurrido el retardo. Inmediatamente obtendrás un mensaje de que faltan datos; ello se debe a que la línea 50 ya no puede leer más datos en una nueva ejecución. Sin embargo, te basta con añadir:

```
260 RESTORE
```

para enviar de nuevo el ordenador al principio de las sentencias DATA, con lo cual puedes hacer correr el programa todas las veces que quieras.

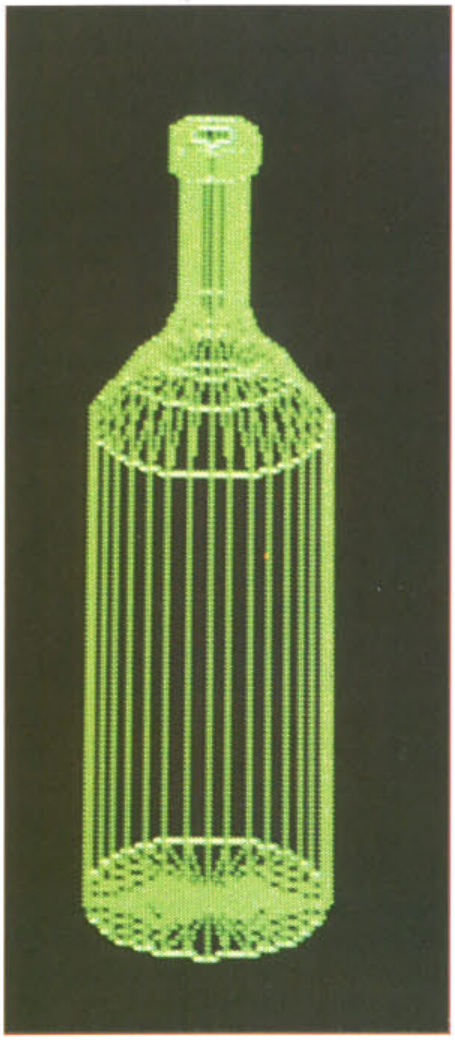
SOLIDOS DE REVOLUCION

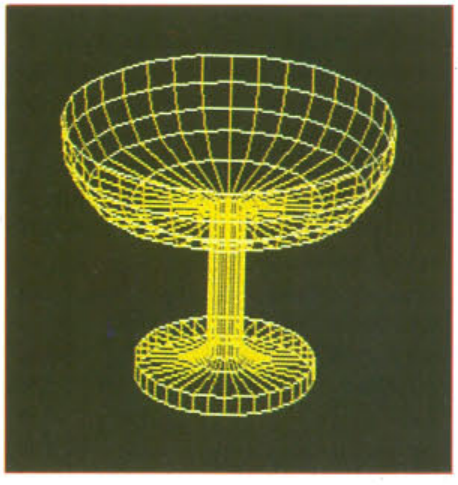
- DIBUJO DE SILUETAS
- «BANDAS DE CAUCHO»
- MODIFICACION DEL PUNTO DE VISTA
- GIRANDO LAS FORMAS
- FUNCIONAMIENTO

Te presentamos aquí una sencilla técnica que te permitirá hacer girar una silueta hasta formar un sólido tridimensional. Puedes utilizarla para diseñar lo que quieras, desde un juego de copas de cóctel hasta un sombrero.

El dibujo de objetos simétricos, especialmente en tres dimensiones, es normalmente muy difícil. Pero con un ordenador se puede conseguir que resulte bastante sencillo. Con el programa que veremos más adelante, lo único que tendrás que hacer es dibujar el contorno de uno de los lados de la forma a dibujar, y el ordenador se ocupará de hacer el resto, llenando el contorno hasta crear un objeto con un entramado de alambres y apariencia de sólido.

El programa funciona efectuando un giro del contorno original alrededor de un eje central. En consecuencia puedes generar cualquier cosa que tenga una sección recta circular, tal como un florero, un jarrón, un vaso, un candelero, una campana, un sombrero, una manzana, una naranja y otras mil cosas más. Debido a que el programa efectúa una rotación del contorno original, la forma sólida producida se llama sólido de rotación o de revolución. El programa te permite además visualizar el objeto desde cualquier ángulo.





DIBUJO DE LA FORMA

Es muy fácil dibujar el contorno de la forma que quieres crear; el programa utiliza para ello la técnica de la banda de caucho, en la que puedes estirar y mover una línea hasta que estés satisfecho con la posición que adopta. La banda de caucho te permite un control interactivo sobre la forma que estás dibujando, pudiendo apreciar a simple vista cuándo está cada línea en su posición correcta.

Puedes dibujar hasta 20 líneas en el contorno. Normalmente esto resulta más que suficiente y casi todos los contornos pueden dibujarse con una media docena de líneas. No obstante las curvas del contorno tienen que formarse a partir de una serie de tramos cortos de rectas lo que requerirá más líneas que un simple contorno angular.

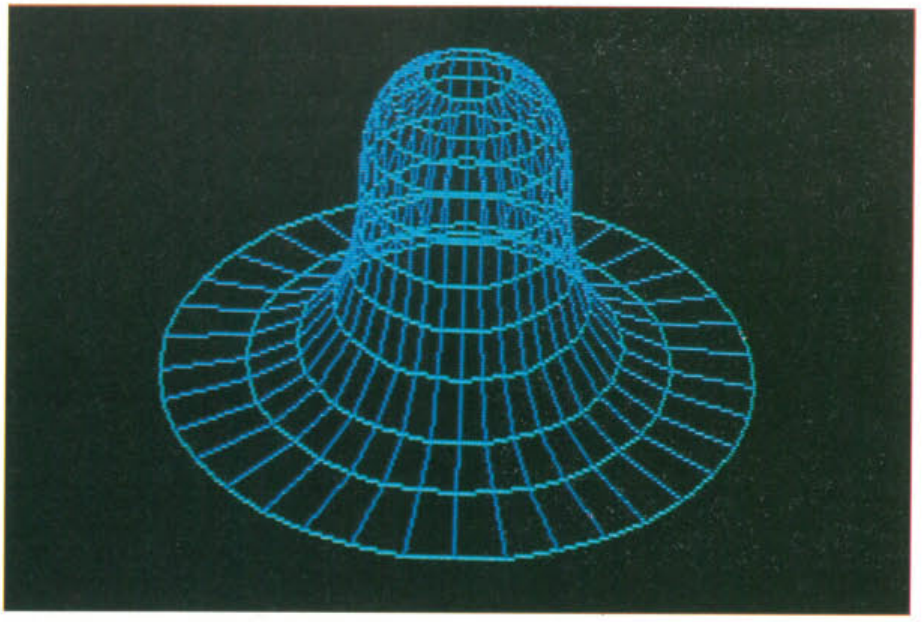
EL PUNTO DE VISTA

El programa también te permite ejercer control sobre la dirección desde la cual se mira al objeto, no sólo desde arriba, abajo o de frente, sino también desde cualquier otro punto, siempre que puedas especificar el ángulo exacto de vista (un ángulo de unos 70 grados te permitirá una buena vista desde arriba, como si el objeto estuviera colocado encima de una mesa). En cualquier momento puedes modificar este ángulo, encargándose el programa de volver a dibujar el sólido de revolución con la nueva orientación.

El programa almacena las coordenadas de todas las líneas. Cuando hayas terminado de dibujarlas, va cogiendo cada línea y la hace girar en torno al punto central en pasos de 18 grados, con un total de 20 pasos para todo el círculo completo. También se toma en cuenta el ángulo del punto de vista, con lo que el círculo aparece cada vez más aplastado a medida que el punto de vista va siendo más bajo. En las imágenes de las páginas que siguen puedes ver el efecto final resultante sobre la pantalla.

Cuando el programa termina de dibujar la primera vista, se queda esperando. Si ahora pulsas la barra espa-

ciadora, puedes elegir la presentación del mismo objeto visto bajo un ángulo diferente.



USO DEL PROGRAMA

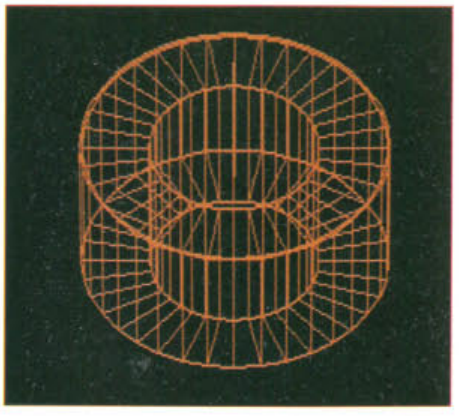
Teclea ya el programa y prueba a dibujar algunas formas. El procedimiento para dibujar consiste en desplazar el cursor al sitio en que quieres comenzar una línea y a continuación pulsar la tecla apropiada para marcar el punto de partida. Seguidamente, mueve el cursor para manipular la línea de la «banda de caucho» sobre la pantalla. Cuando estés satisfecho con la posición alcanzada y quieras «fijarla» en su sitio, tienes que pulsar otra tecla. Continúa con este mismo proceso, moviendo y estirando la banda de caucho y fijando líneas, hasta construir todo el perfil.

Para mover el cursor, tienes que utilizar las teclas [Z], [X], [;] y [/], el símbolo ← para marcar la posición de partida y [SPACE] para «fijar» cada línea. Cuando hayas terminado, pulsa [RETURN].

Cuando el dibujo ya esté completo, pulsa Q para pasar a la siguiente página, la ejecución del programa.

Al dibujar las formas puedes acelerar el movimiento del cursor manteniendo pulsada al mismo tiempo la tecla [SHIFT].

```
1 REM**ESTE PROGRAMA PRECISA
  EL SIMON'S BASIC
10 HIRES 3,6:COLOUR 6,6:DIM
   A(19,1):POKE 650,128
20 RD=ATN(1)/45
40 BX=141:BY=190:XX=BX:YY=BY
   :X=BX:Y=BY
50 REC 59,141,163,51,2
60 LINE BX,BY,Y,2
70 GET K$:CP=PEEK(197):IF K$
   =" " THEN GOSUB 500
80 M=2:IF PEEK(653)=1 THEN
   M=8
90 LINE BX,BY,X,Y,2:IF K$=
   "[FLECHA izq.]"AND A$=""
   THEN BX=X:BY=Y:BF=1:GOSUB
   500
100 IF K$=CHR$(13) AND LEN
    (A$)>0 THEN 160
110 IF CP=50 AND Y-M>142
    THEN Y=Y-M
120 IF CP=55 AND Y+M<191
    THEN Y=Y+M
130 IF CP=12 AND X-M>140
    THEN Y=X-M
140 IF CP=23 AND X+M<220
    THEN X=X+M
150 GOTO 60
160 GOSUB 2000:HIRES 0,0:
    MULTI 3,6,14:COLOUR 6,0
170 GOSUB 1000:COLOUR 14,0
190 GET K$:IF K$=" "THEN 160
195 GOTO 190
500 LINE BX,BY,X,Y,1
510 LINE 281-BX,BY,281-X,Y,1
520 BX=X:BY=Y:A$=A$+CHR$(X-
    141)+CHR$(190-Y):RETURN
1000 FOR B=1 TO LEN(A$)STEP2
1010 X=ASC(MID$(A$,B,1)):Y=
     ASC(MID$(A$,B+1,1))
1020 GOSUB 1500:NEXT B
```



```
1040 RETURN
1500 FOR C=0 TO 399 STEP 20
1510 D=C:PY=IS*(Y*5)
1520 GOSUB 1530:NEXTC:RETURN
1530 YD=100-SIN(D*RD)*X*COS
     (I*RD)-PY:XD=80+COS(D*
     RD)*X
1540 IF C=0 THEN BX=XD:BY=YD
     :XX=XD:YY=YD
1550 LINE XX,YY,XD,YD,1:XX=
     XD:YY=YD
1560 IF B=1 AND BF=1 THEN
     1580
1565 IF B=1 THEN LINE 80,91,
     XD,YD,2:GOTO 1580
```

```
1570 LINE A(C/20,0),A(C/20,1
     ),XD,YD,3
1580 XX=XD:YY=YD:A(C/20,0)=
     XD:A(C/20,1)=YD:RETURN
2000 NRM:I=0:INPUT"[CTRL+4]
     [SHIFT+CLR/HOME]ANGULO
     DE VISTA (0-180)";I
2010 IF I<0 OR I>180 THEN
     2000
2020 IS=SIN(I*RD):RETURN
```

dibujar el contorno, crear el sólido y hacer que éste se ponga a girar.

INTRODUCCION DEL PERFIL

La rutina de la banda de caucho comienza en la línea 60. Fundamentalmente esta rutina se ocupa de comprobar qué teclas han sido pulsadas y de actualizar las coordenadas de la línea estirada. El dibujo real de la línea se hace por medio de la rutina de la línea 500, que también se ocupa de dibujar la imagen especular de dicha línea.

Cuando una línea es fijada, se almacenan las coordenadas de su extremo. Antes que nada, se resta un número de cada coordenada para tener un punto medido con relación a la línea central. A continuación se cargan dichos números en una cadena de caracteres que se añade al extremo de A$.

Con la rutina de la línea 2000 puedes introducir el ángulo del punto de vista; se acepta cualquier ángulo entre 0 y 180 grados. A continuación el programa salta a la rutina de la línea 1000 que se encarga de crear y dibujar el sólido tridimensional.

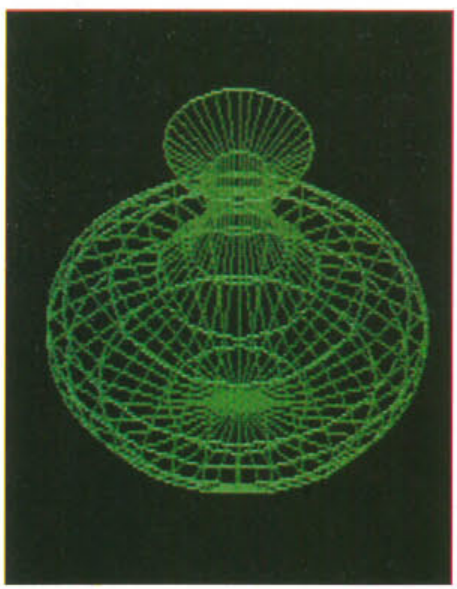
LA TERCERA DIMENSION

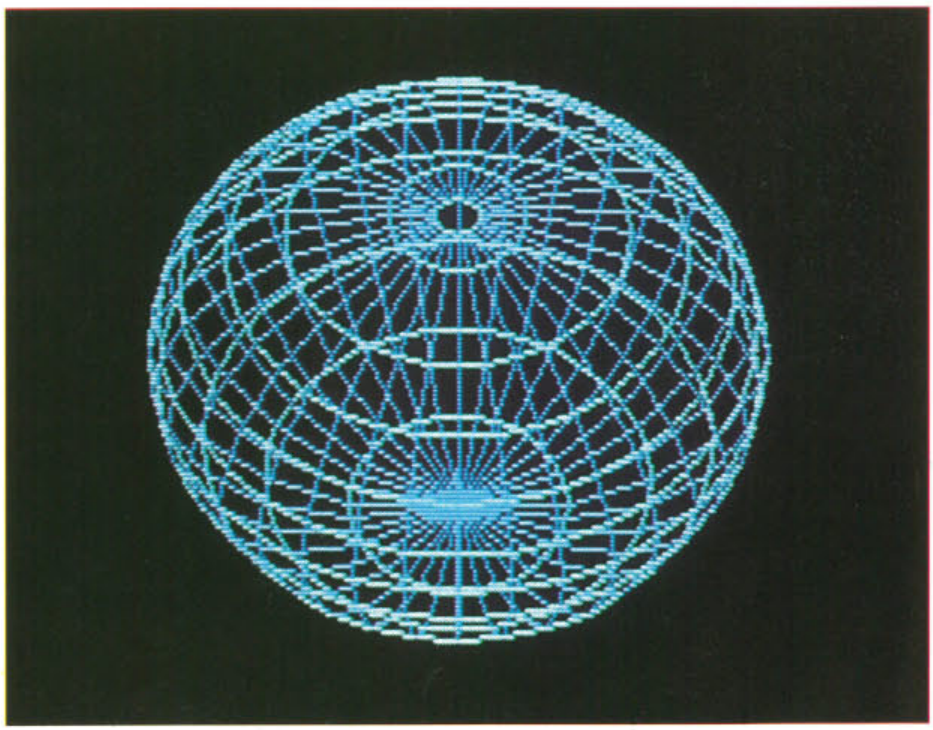
La siguiente rutina utiliza las líneas que has dibujado y el ángulo de vista introducido para transformar el contorno en un sólido de aspecto tridimensional.

En la línea 1000 hay un bucle controlado por la variable B, que va seleccionando cada una de las líneas y tomando sus coordenadas terminales de la cadena a$. Seguidamente salta a la siguiente rutina, que comienza en la línea 1500.

Esta rutina desplaza las coordenadas de manera que tengan en cuenta el punto de vista seleccionado por tí y a continuación dibuja la línea que utiliza estas coordenadas desplazadas. Este proceso se repite 20 veces, de modo que tu línea original se repite en 20 posiciones diferentes alrededor de un círculo. La rutina vuelve después a la línea 1000 para hacer lo mismo con las siguientes líneas. Tomando las líneas de una en una, la imagen va siendo construída por pasos, como podrás comprobar al ejecutar el programa.

Si observas el programa cuidadosamente, podrás ver que el procedimiento para dibujar la primera de tus líneas

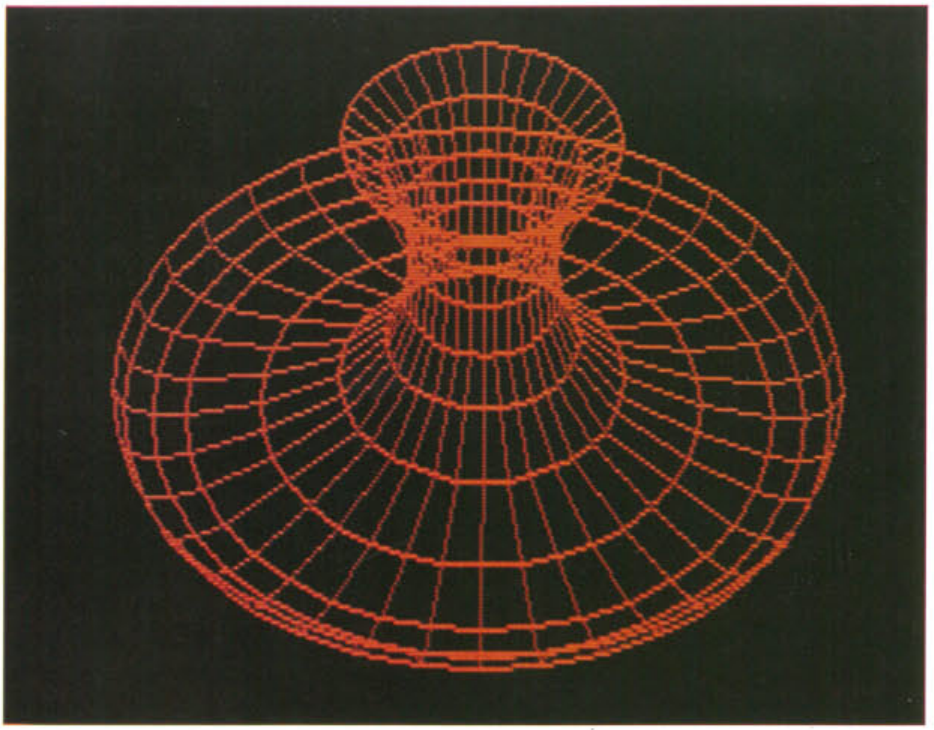




FUNCIONAMIENTO

La parte más importante del programa son las rutinas que te permiten

es ligeramente diferente del seguido para las demás líneas. Esto se debe a que las coordenadas desplazadas que va calculando siempre corresponden al final de las líneas, mientras que las coordenadas del comienzo son los finales de las líneas anteriores. Sin embargo en la primera línea hay que calcular también las coordenadas de comienzo.



Las líneas 1560 y 1565 sirven para comprobar si se está trabajando sobre la primera línea. Cuando es B = 1, significa que se trata de la primera línea, mientras que cuando es BF = 1, significa que la primera línea se ha desplazado un espacio. Con el desplazamiento de un espacio, lo que sucede es que se calculan las coordenadas y se almacenan en la matriz A(). Estas coordenadas formarán entonces el comienzo del siguiente conjunto de líneas. Si el primer movimiento no corresponde a un blanco, significa que se ha empezado a partir del centro por lo que los «rayos» se dibujan a partir del centro.

De esta forma, para cada línea de tu dibujo se dibuja un anillo completo de líneas, cuyos extremos quedan unidos por un círculo, produciendo un efecto tridimensional. Las coordenadas de cada punto terminal se almacenan en la matriz A() de forma que el ordenador conoce las posiciones de partida del siguiente anillo.

17 % de descuento

Por sólo 290 Ptas. ejemplar, y recibidos todos cómodamente en su hogar...

¡¡Suscríbase ahora a INPUT!!

commodore

PRECIO DE CUBIERTA PTAS. 350
MENOS:
17% de descuento al suscriptor PTAS. (60)
USTED PAGA SOLO PTAS. 290 POR EJEMPLAR

Entrega a domicilio GRATIS

SUSCRIPCION ANUAL = 11 EJEMPLARES
3.850 Ptas.
(660 Ptas.)
Usted paga sólo
3.190 Ptas.

INPUT le proporciona
INFORMACION... DIVERSION...
...FORMACION (un curso completo de programación)...
...LA POSIBILIDAD DE MEJORAR su NIVEL PROFESIONAL...
EL NIVEL DE LOS ESTUDIOS...

...Descubra el mundo de la informática...
...Aprenda a programar con facilidad...
...Diviértase con los ordenadores...
...Esté siempre al día...

Recorte y envíe este cupón de inmediato a EDISA, López de Hoyos, 141-28002 Madrid, o bien llámenos al Telf. (91) 415 97 12

BOLETIN DE SUSCRIPCION

SI, envíenme INPUT COMMODORE durante 1 año (10 ejemplares + el extraordinario de verano), al precio especial de oferta de 3.190 Ptas. AHORRANDOME 660 Ptas. sobre el precio normal de portada de 11 ejemplares sueltos. (Por favor cumplimente este boletín con sus datos personales e indíquenos con una (X) la forma de pago por usted elegida, mételo en un sobre y deposítelo en el buzón más próximo).

NOMBRE ..... APELLIDOS .....
DOMICILIO ..... NUM ..... PISO ..... ESCALERA ..... COD. POSTAL .....
POBLACION ..... PROVINCIA ..... TELF .....
PROFESION .....

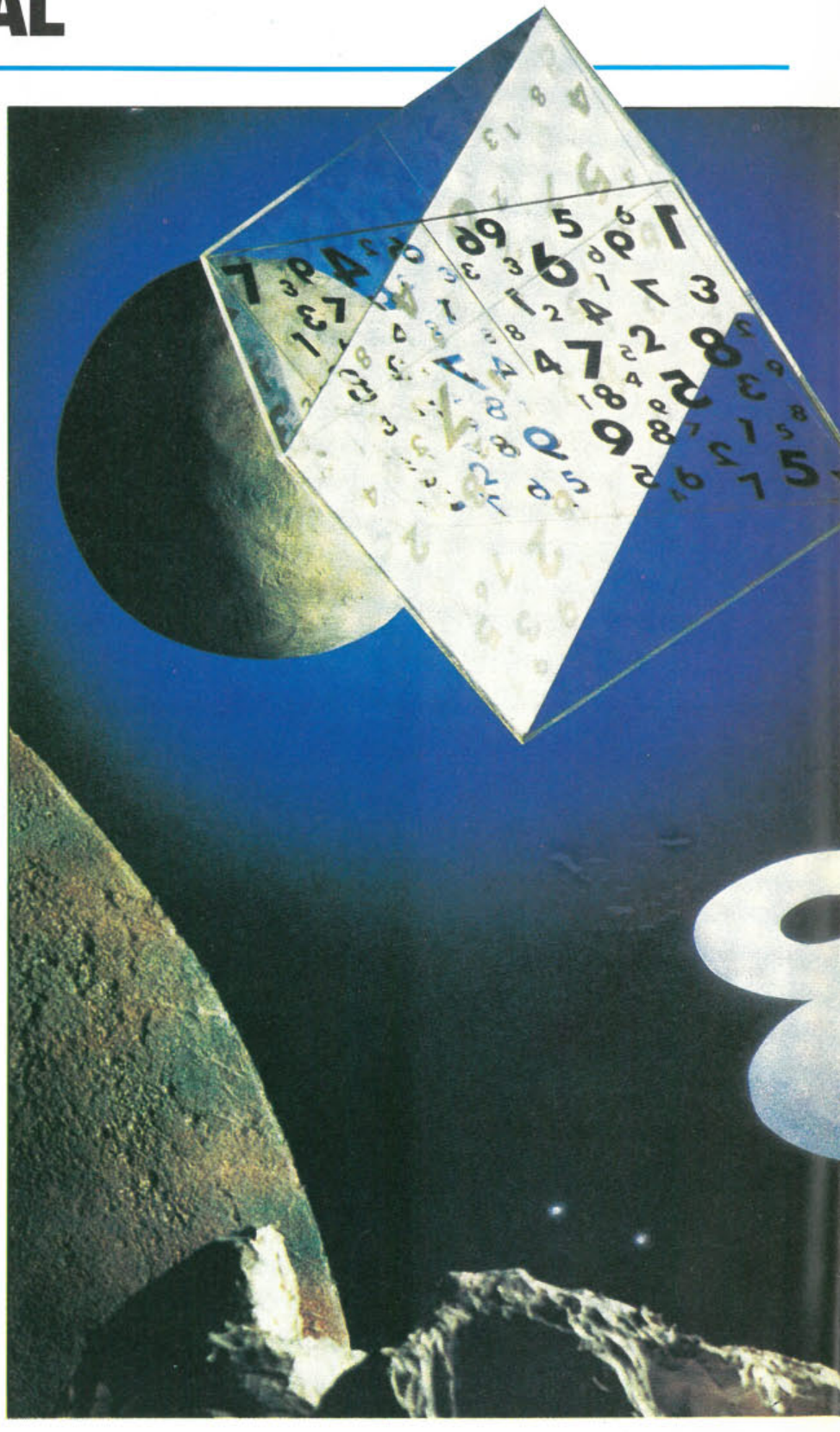
FORMA DE PAGO ELEGIDA: Reembolso ☐ Domiciliación Bancaria ☐
Talón nominativo que adjunto a favor de EDISA ☐

INSTRUCCIONES DE DOMICILIACION BANCARIA (si es elegida por usted)

Muy señores míos: ..... de ..... de 19....
Les ruego que, con cargo a mi cuenta nº ..... atiendan, hasta nuevo aviso, el pago de los recibos que les presentará Editorial PLANETA-AGOSTINI a nombre de: .....
..... BANCO/C. de AHORROS .....
..... DIRECCION .....
FIRMA

MANEJO DE LA ARITMETICA HEXADECIMAL

Se puede aprender a contar con los dedos y aprender a contar en base 16. Pero aunque tú no tengas 16 dedos, encontrarás el manejo de cifras hexadecimales mucho más sencillo que tener que vértelas con grandes ristras de ceros y unos.



Los ordenadores, en lo más profundo de sus circuitos ejecutan toda su aritmética en binario, utilizando un sistema de números compuesto totalmente por ceros y unos, lo cual crea ciertas dificultades a los operadores humanos.

En cuanto los números alcanzan un tamaño moderado, enseguida te enfrentas con más roscos que una fábrica de *donuts*. Y no resulta fácil teclear una serie larga de ceros y unos. Es muy fácil cometer un error y es muy difícil detectarlo después.

La manera que tiene el operador para evitar estas dificultades es utilizar una base de numeración diferente recurriendo, entonces, a la numeración hexadecimal.

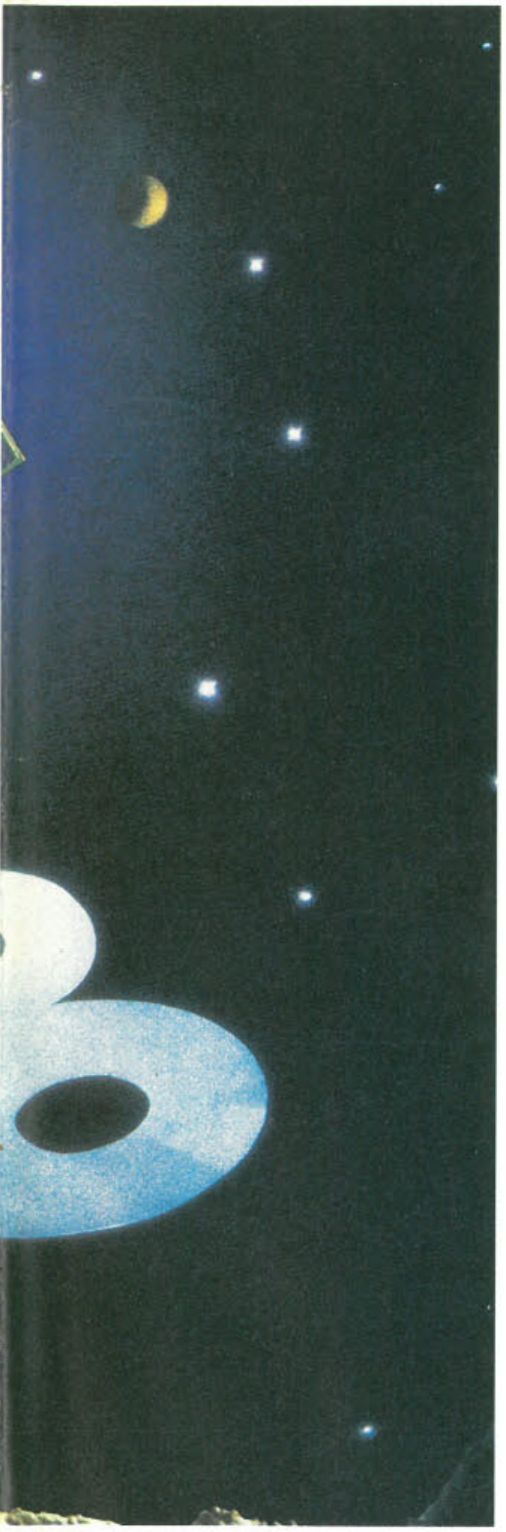
Los números hexadecimales son los que utilizan como base de numeración el número 16. Son lo suficientemente próximos a los números de base 10 como para resultar de manejo relativamente sencillo por parte de un operador humano.

Además 16 es igual a 2 × 2 × 2 × 2, lo cual significa que la conversión entre binario y hexadecimal es simple. El número decimal 16 se escribe 10 en hexadecimal y 10000 en binario. Además, todo número entre el 0 y el 15 se representa mediante un número binario de cuatro dígitos.

Para poder utilizar un sistema de numeración con una base mayor que 10, hay que definir símbolos para los nuevos dígitos.

En hexadecimal, 10 se representa por A, once por B, doce por C, trece por D, catorce por E y quince por F.

- POR QUE SE UTILIZAN
- LOS NUMEROS HEXADECIMALES
- RELACION ENTRE BINARIO Y HEXADECIMAL
- CONVERSION



CONVERSION DE BINARIO A HEXADECIMAL

La conversión a hexadecimal de los números binarios de ocho dígitos que utilizan los ordenadores domésticos, es particularmente fácil. Tienes que dividir el número binario comprendido entre $0000 y $FFFF en dos secciones de ocho dígitos. A continuación, los primeros cuatro dígitos se convierten directamente en un dígito hexadecimal, y los cuatro restantes se convierten en otro dígito.

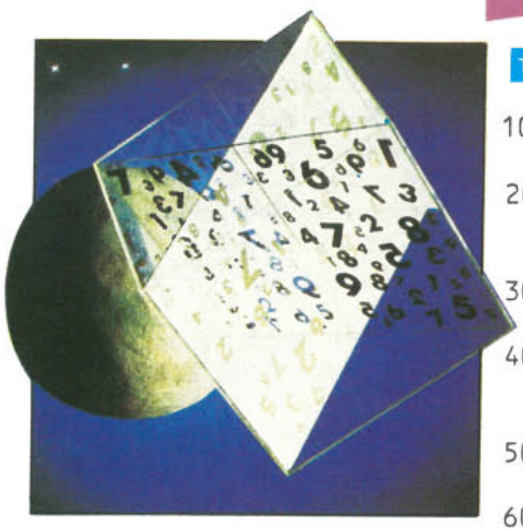
La conversión de decimal a hexadecimal es más difícil. Para hacerla tienes que hacer divisiones sucesivas del número decimal por 16. Los restos resultantes en cada división son los dígitos hexadecimales buscados.

Por ejemplo, al dividir 1226 por 16, obtienes como cociente 76 y de resto 10. En hexadecimal 10 se escribe A. 76 dividido por 16 da cuatro y de resto 12, que en hexadecimal se escribe C. Por último, 4 dividido entre 16 da 0 y de resto 4. En consecuencia 1226 en decimal se escribe 4CA en hexadecimal. Es habitual que para distinguir un número hexadecimal se le ponga el signo $ a la izquierda.

El siguiente programa es bastante largo, pero merece la pena teclearlo, porque te ayudará a asentar en tu mente la forma en que se opera esta conversión:

Teclea para Commodore-64

```
20 PRINT"[SHIFT+CLR/HOME]"
   CHR$(8):FOR Z=1 TO 8:READ
   A(Z):NEXT Z:DATA 128,64,
   32,16,8,4,2,1
30 K$="0123456789ABCDEF":
   POKE 650,255:POKE 53280,0
   :POKE 53281,0
40 PRINT"[CLR/HOME]"TAB(13)
   [CTRL+8]"BIN,DEC,HEX"
50 PRINT TAB(13)"[CTRL+2]
   ----------"
60 PRINT"[3*CRSR abajo]
   [CTRL+8]BINARIO[CTRL+4]:
   [CTRL abajo]"
70 PRINT"[3*CRS abajo]
   [CTRL+8]DECIMAL[CTRL+4]:
   [CTRL abajo]"
80 PRINT"[3*CRSR abajo]
   [CTRL+8]HEXADECIMAL
   [CTRL+4]:[CRSR abajo]"
90 PRINT"[CLR/HOME][6*CRSR
   abajo]":V$="":FOR Z=1 TO
   8:V$=V$+LEFT$(" "+STR$
   (B(Z))+"[2*ESPACIO]",4):
   NEXT Z
100 PRINT MID$(V$,3,14)
   "[2*ESPACIO]"RIGHT$
   (V$,16)
110 PRINT"[3*CRSR abajo]":
   V$="":FOR Z=1 TO 8
120 V$=V$+LEFT$("+"+RIGHT$
   (STR$(D(Z)),LEN(STR$
   (D(Z)))-1)+"[3*ESPACIO]"
   ,4):NEXT Z
130 PRINT MID$(V$,2,15);
   "[3*ESPACIO]";MID$
   (V$,18,16)"=[4*ESPACIO]
   [4*CRSR izq.]";
140 PRINT D(1)+D(2)+D(3)+
   D(4)+D(5)+D(6)+D(7)+D(8)
150 PRINT "[3*CRSR abajo]":
   V$="":FOR Z=1 TO 8
160 V$=V$+LEFT$("+"+RIGHT$
   (STR$(H(Z)),LEN(STR$
   (H(Z)))-1)+"[3*ESPACIO]"
   ,4):NEXT Z
170 X1$=MID$(K$,H(1)+H(2)+
   H(3)+H(4)+1,1):X2$=MID$
   (K$,H(5)+H(6)+H(7)+H(8)
   +1,1)
180 PRINT MID$(V$,2,15);
   [CRSR izq.]=[CTRL+3]"X1$
   "[CTRL+4][2*ESPACIO]";
   MID$(V$,18,16);"= [CTRL
   +3]"X2$
190 PRINT"[2*CRSR abajo]
   [CTRL+2]HEX TOTAL :
   [6*ESPACIO][5*CRSR izq.]
   [CTRL+3]";X1$+X2$
195 PRINT"<ESPACIO> INCREMEN
   TA EL NUMERO"
196 PRINT"<D> DECREMENTA EL
  NUMERO"
200 GET A$:IF A$="" THEN 200
210 POKE 198,0:IF A$=" "THEN
   A=A+1:IF A>255 THEN A=0
220 IF A$="D" THEN A=A-1:IF
   A<0 THEN A=255
230 IF A$<>" " AND A$<>"D"
   THEN 350
240 AA=A
```



```
250 FOR Z=1 TO 8:IF A(Z)<=AA
    THEN B(Z)=1:D(Z)=A(Z):AA
    =AA-A(Z):GOTO 270
260 B(Z)=0:D(Z)=0
270 NEXT
280 FOR Z=1 TO 4:IF B(Z)=1
    THEN H(Z)=A(Z+4):
    GOTO 300
290 H(Z)=0
300 NEXT
310 FOR Z=1 TO 4:IF B(Z+4)=1
    THEN H(Z+4)=A(Z+4):
    GOTO 330
320 H(Z+4)=0
330 NEXT Z
340 GOTO 90
350 I$="":PRINT"[CTRL+2]
    [CLR/HOME][22*CRSR
    abajo] ESCRIBE EL NUMERO
    (0-255)?:[4*ESPACIO]
    [3*CRSR izq.]";
360 FOR Z=1 TO 3
370 GET J$:PRINT"*[CRSR
    izq.] [ESPACIO]LCRSR
    izq.]";:IF J$="" THEN 370
380 IF J$=CHR$(13) THEN 440
390 IF J$=CHR$(20) THEN 350
400 IF ASC(J$)<48 OR ASC(J$)
    > 57 THEN 370
410 I$=I$+J$:PRINT J$;:
    NEXT Z
420 GET J$:IF J$=CHR$(20)
    THEN 350
430 IF J$<>CHR$(13) THEN 420
440 IF VAL(I$)<0 OR VAL(I$)>
    255 THEN 350
450 PRINT:PRINT"[CRSR
    arriba][40*ESPACIO]"
460 A=VAL(I$):GOTO 240
```

Teclea para Vic-20

```
10 PRINT "SHIFT+CLR/HOME]"
   CHR$(8)
20 FOR Z=1 TO 8:READ A(Z):
   NEXT:DATA 128,64,32,16,
   8,4,2,1
30 K$="0123456789ABCDEF":
   POKE 650,255:POKE 36879,8
40 PRINT "[CLR/HOME]
   [5*ESPACIO][CTRL+8]BIN,
   DEC,HEX"
50 PRINT"[5*ESPACIO][CTRL+2]
   ----------"
60 PRINT"[2*CRSR abajo]
   [CTRL+8]BINARIO[CTRL+4]:"
70 PRINT"[3*CRSR abajo]
   [CTRL+8]DECIMAL[CTRL+4]:"
80 PRINT "[3*CRSR abajo]HEXA
   DECIMAL[CTRL+4]:"
90 PRINT"[CLR/HOME][5*CRSR
   abajo][CTRL+4]":V$="":FOR
   Z=1 TO 8:V$=V$+STR$(B(Z))
   :NEXT Z
100 PRINT LEFT$(V$,8)
    "[3*ESPACIO]"RIGHT$(V$,8)
110 PRINT"[2*CRSR abajo]":V$
    ="":FOR Z=1 TO 8
120 V$=V$+LEFT$("+"+RIGHT$
    (STR$(D(Z)),LEN(STR$(D
    (Z)))-1)+"[3*ESPACIO]"
    ,4):NEXT Z
130 PRINT MID$(V$,1,15):
    PRINT MID$(V$,17,16)
    "[CRSR izq.]=[4*ESPACIO]
    [4*CRSR izq.]";
140 PRINT D(1)+D(2)+D(3)+
    D(4)+D(5)+D(6)+D(7)+D(8)
150 PRINT "[CRSR abajo]":V$=
    "":FOR Z=1 TO 8
160 V$=V$+"+"+RIGHT$(STR$(H
    (Z)),1):NEXT Z
170 X1$=MID$(K$,H(1)+H(2)+
    H(3)+H(4)+H(5)+H(6)+H(7)
    +H(8)+1,1)
180 PRINT LEFT$(V$,8):"=
    [CTRL+3]"X1$;" [CTRL+4]"
    RIGHT$(V$,8)"=[CTRL+3]"
    X2$
190 PRINT"[CRSR abajo][CTRL
    +2]TOTAL HEX:[6*ESPACIO]
    [5*CRSR izq.][CTRL+3]";
    X1$+X2$
200 GET A$:IF A$="" THEN 200
210 POKE 198,0:IF A$=" "
    THEN A=A+1:IF A>255 THEN
    A=0:REM**<ESPACIO>INCRE
    MENTA NUMERO
220 IF A$="D" THEN A=A-1:IF
    A<0 THEN A=255:REM**<D>
    DECREMENTA NUMERO
230 IF A$<>" " AND A$<>"D"
    THEN 350
240 AA=A
250 FOR Z=1 TO 8:IF A(Z)<=AA
    THEN B(Z)=1:D(Z)=A(Z):AA
    =AA-A(Z):GOTO 270
260 B(Z)=0:D(Z)=0
270 NEXT
```

P y R

¿Cómo se hace la conversión de hexadecimal a decimal?

Cada dígito sucesivo de un número expresado en forma hexadecimal tiene un peso 16 veces superior que el dígito que queda a su derecha. Así, para convertir en decimal un número como $F6DA, se toma el número de la derecha y se convierte a decimal. Como es A, su valor es 10. El siguiente dígito de la izquierda tiene un peso 16 veces mayor, por lo que hay que pasarlo a decimal y multiplicarlo por 16. Al ser el número D, que vale 13, tenemos 13×16= 208. El siguiente dígito por la izquierda tiene un peso 16 veces mayor, o sea, 6×16×16= 1.536. El último dígito de la izquierda ha de multiplicarse aún por 16 una vez más. Como es el número F, que vale 15, tenemos: 15 × 16 × 16 × 16= 61.440. Así el número $F6DA en hexadecimal será en decimal 10+208+1.536+ 61.440= 63.194. También puedes utilizar el programa anterior para convertir los dos números hexadecimales al mismo tiempo y a continuación multiplicar el de la izquierda por 256.

```
280 FOR Z= 1 TO 4:IF B(Z)=1
    THEN H(Z)=A(Z+4):GOTO 300
290 H(Z)=0
300 NEXT Z
310 FOR Z=1 TO 4:IF B(Z+4)=1
    THEN H(Z+4)=A(Z+4):
    GOTO 330
320 H(Z+4)=0
330 NEXT Z
340 GOTO 90
350 I$="":PRINT "[CTRL+6]
    [CLR/HOME][19*CRSR abajo]
    ESCRIBE EL NUMERO?
    [CTRL+2](0-255)[CTRL+5]>
    [4*ESPACIO][3*CRSR izq.]
    ";
360 FOR Z=1 TO 3
370 GET J$:PRINT"*[CRSR izq.]
    [ESPACIO][CRSR izq.]";:
    IF J$="" THEN 370
380 IF J$=CHR$(13) THEN 440
390 IF J$=CHR$(20) THEN 350
400 IF ASC(J$)<48 OR ASC(J$)
    >57 THEN 370
410 I$=I$+J$:PRINT J$;:NEXT Z
420 GET J$:IF J$=CHR$(29)
    THEN 350
430 IF J$<>CHR$(13) THEN 420
440 IF VAL(I$)<0 OR VAL(I$)>
    255 THEN 350
450 PRINT:PRINT"[2*CRSR
    arriba][28*ESPACIO]";
460 A=VAL(I$):GOTO 240
```

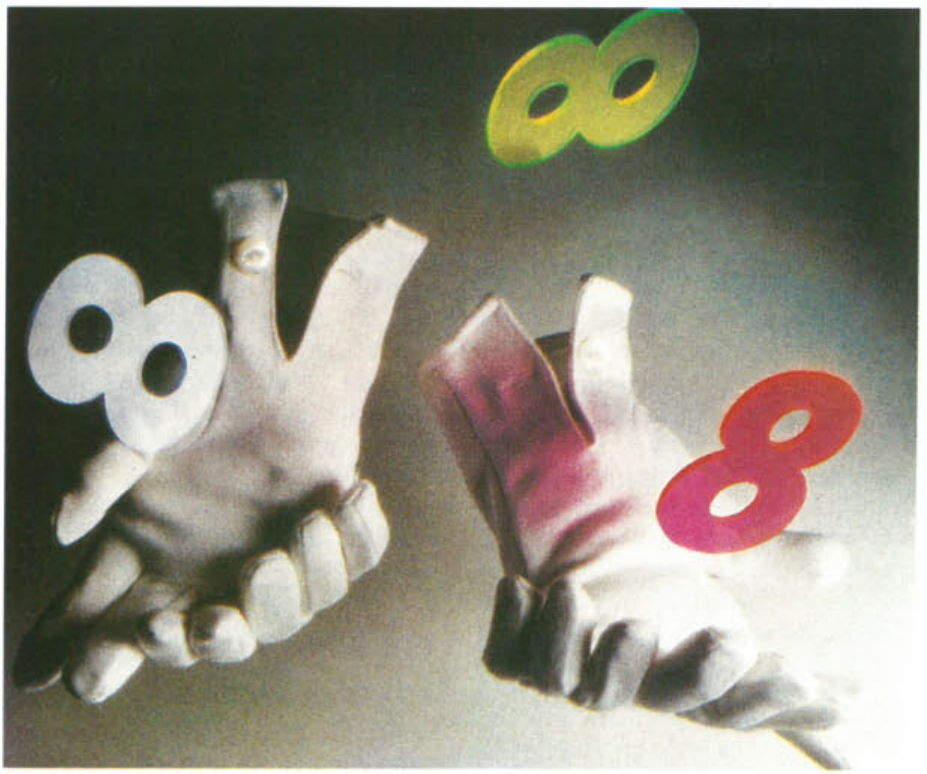
Cuando hayas tecleado todo el programa y lo ejecutes, observarás que se ponen a cero todos los números: binarios, decimales y hexadecimales. Si aprietas la tecla espaciadora aparecerá un 1 en cada base. Sigue apretando y el ordenador se pondrá a contar, sumando cada vez 1 a cada uno de los totales. Observa que el equivalente decimal se calcula sumando el valor correspondiente a todo lugar que en binario tiene un 1.

El valor hexadecimal se calcula haciendo exactamente lo mismo, salvo que se toman cuatro dígitos a la vez.

Al pulsar la tecla [B], se restará 1 de cada uno de los números y se ejecutará el programa.

Puedes almacenar este programa y utilizarlo para convertir en hexadecimal números binarios siempre que te haga falta. La forma rápida de hacer esta conversión es pulsar cualquier tecla, con excepción de la barra de espacio o la [B]. En la pantalla aparecerá un signo de interrogación. Introduce cualquier número decimal que sea menor que 255, pulsa [RETURN] y te aparecerá su equivalente en binario y en hexadecimal.

Observarás que el número mayor que puede ser representado por un número de ocho bits en binario, es decir por un byte, es 11111111. Su equivalente decimal es 255, que en hexadecimal se escribe 255. Este es el mayor número posible de dos dígitos en hexadecimal. Cualquier número almacenado en un byte de la memoria de tu ordenador puede representarse en hexadecimal por medio de dos dígitos. Y todo el lenguaje máquina está construído enteramente a base de estos números hexadecimales de dos dígitos.



NUMEROS MAS GRANDES

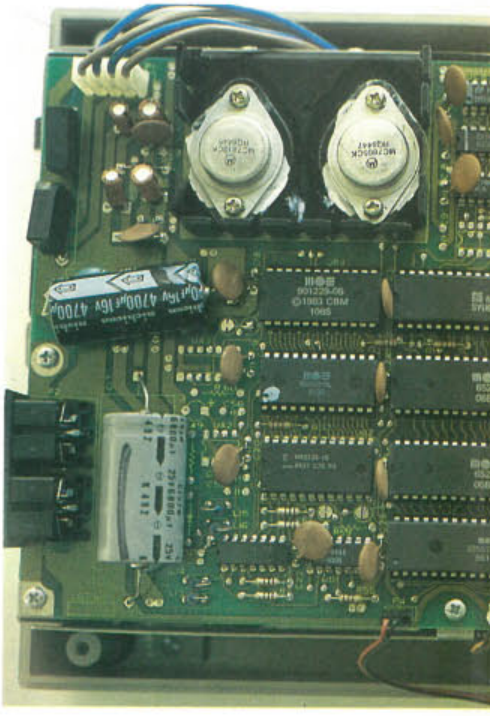
Tu ordenador es capaz de manejar números mayores que 255, por el sencillo método de fraccionarlos en dos partes y almacenarlas en dos posiciones adyacentes de memoria. Esto te permitirá almacenar cualquier número hasta FFFF en hexadecimal, o lo que es igual, 65535 en decimal. $FFFF es un número muy importante en los ordenadores domésticos, ya que es el máximo número de posiciones de memoria que se pueden direccionar.

Se pueden almacenar en memoria números aún mayores, fraccionándolos en tres o cuatro bytes y almacenándolos en direcciones de memoria sucesivas. En cuanto a la forma de almacenar dichos bytes, es una cuestión de convenio. En tu máquina se almacena el byte más bajo en la posición de memoria más baja y el byte alto en la posición más alta de memoria.

NO OLVIDES EL TELEFONO...

Cuando, por cualquier motivo, nos escribas.

LA 1541 FANTASTICA



¿Cuántas veces habrás despegado la etiqueta de protección en un disco que creías tener cerrado? ¿Cuántas has maldicho el momento al comprobar que la etiqueta de protección ya no pega como antes y produce atascos en la unidad de discos? También has quedado maravillado por esos discos que vienen grabados sin que estos lleven la ranura que permite la grabación.

Pues bien aquí tienes la solución. Con este montaje podrás escribir sobre discos que tengan la etiqueta de protección, incluso podrás formatear y escribir sobre la otra cara del disco sin necesidad de abrir la ranura de escritura, ante el asombro de propios y extraños (entiéndase amigos). ¡La de etiquetas que nos ahorramos!.

Bien, la explicación es bien sencilla, en la unidad de discos **1541** la detección de la ranura se efectúa a través de un LED y un fotodiodo que se encuentran alojados en la guía izquierda de la unidad. Cuando la ranura del disco está abierta, la luz emitida por el LED es captada por el fotodiodo, haciendo que éste disminuya la corriente de paso y presentando un nivel bajo a la entrada de una puerta lógica que informa a la VIA 2 (**6522**) de ello, en cambio si la ranura del disco está tapada, o no existe, la luz del LED jamás alcanzará al fotodiodo y éste mantendrá un nivel alto en la entrada de la puerta, indicando que el disco no puede ser escrito.

Para los curiosos veamos en la tabla 1 cómo se comporta la unidad en parte de una rutina encargada de formatear el disco.

Como vemos no se puede enmascarar la detección del fotodiodo mediante *software*. ¿Cual es entonces la solución a nuestro problema? Está claro que la intención es engañar al fotodiodo para que este crea estar detectando luz del LED cuando a nosotros nos interese, o mejor dicho, simularemos esta acción.

Para ello sólamente tendremos que cortocircuitar a voluntad el fotodiodo y de esta manera su «resistencia aparente» será CERO y por lo tanto mandaremos un nivel bajo a la puerta en cuestión, simulando de esta manera la detección consiguiendo así nuestro propósito.

¿Que, cómo hacemos esto? Pues bien sencillo, nos armamos con un soldador, dos trozos de cable fino, un poco de habilidad y un pulsador o interruptor miniatura. Aclaremos por qué usar uno u otro, si utilizamos un pulsador, cada vez que escribamos en el disco protegido, tendremos que mantenerlo constantemente pulsado, mientras que si empleamos un interruptor esto no será necesario, pero si lo dejamos activado por descuido corremos el riesgo de escribir o borrar en un disco con protección, en el que no nos interesaba hacerlo. Por este motivo aconsejamos la instalación de un pulsador, que aunque debemos pulsarlo durante la grabación, es más seguro. Aclarado esto continuamos con el montaje, procedemos entonces a la apertura de nuestra amadisima unidad de disco y localizamos el conector 'P6', que se encuentra en la tarjeta principal, conforme se mira la unidad de frente es el **tercero**, este conector es el más grande y tiene 15 patillas, se cuentan de atrás hacia adelante, es decir desde la entrada serie hacia la trampilla del disco (ver Fig. 1). Este conector esta distribuido de la siguiente manera (ver tabla 2):

Nuestra atención debe centrarse en las patillas 12 y 13, que son las que van al fotodiodo, y las que interesan cortocircuitar. Asi que pondremos la unidad de lado y soldaremos dos cables en las soldaduras que se encuentran debajo de la placa correspondien-

```
TABLA 1
FB00  AD 00 1C  LDA $1C00 ;Carga en el acumulador el
                           byte correspondiente a la
                           lectura de la VIA 6522/2
FB03     29 10  AND #$10  ;Realiza un AND del acumula
                           dor con $10 que es el bit
                           encargado del fotodiodo
FB05     D0 05  BN3 $FB0C ;Salta y continua en $FB0C si
                           no hay proteccion contra
                           escritura
FB07     A9 08  LDA #$08  ;Como si hay proteccion carga
                           en el acumulador el tipo de
                           error $08
FB09  4C D3 FD  JMP $FDD3 ;Salta a la subrutina encar-
                           gada de escribir el tipo de
                           error 26, 'write protetion'.
FB0C  20 A3 FD  JRS $FDA3 ;Continua el programa...
```

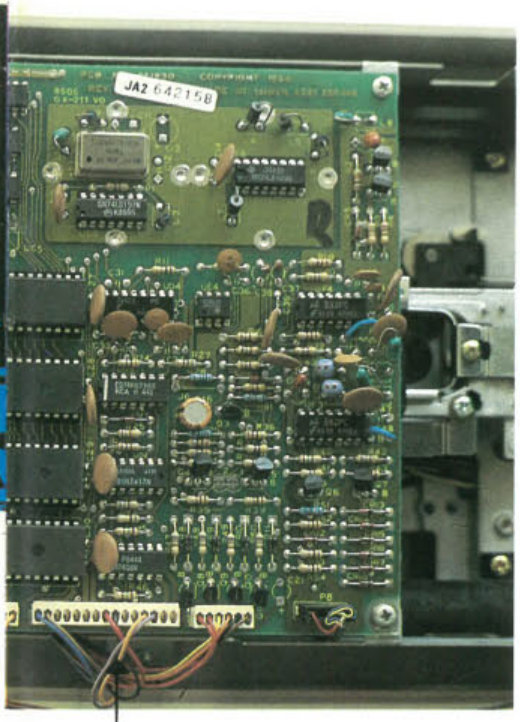
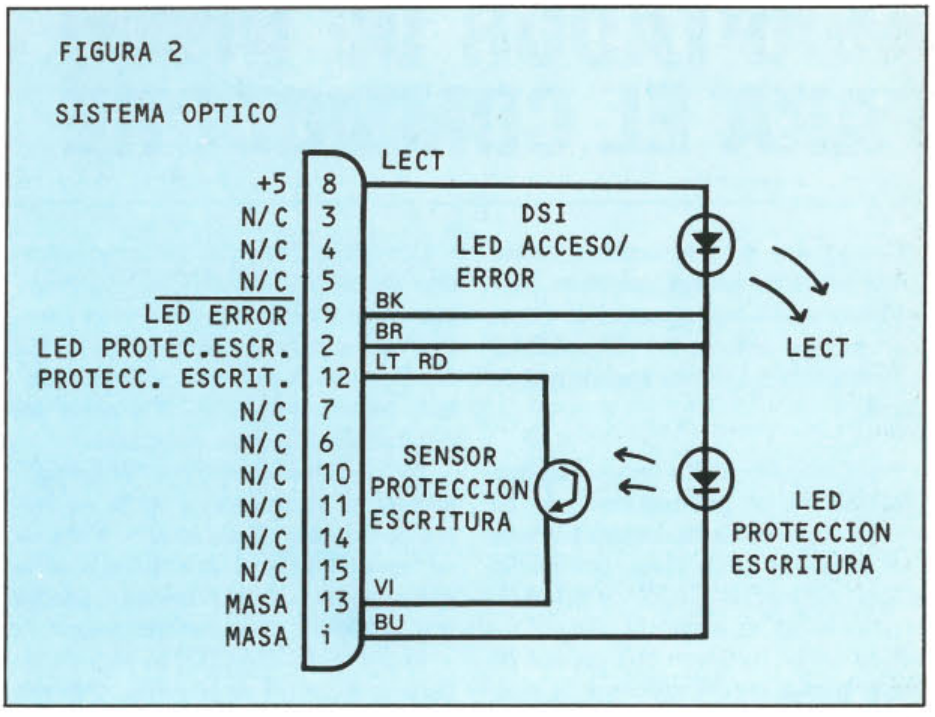


FIGURA 1



tes a las patillas 12 y 13, una vez terminado esto comprobaremos que funciona uniendo las puntas de los cables, dando tensión a la unidad y mandando formatear una cara de un disco que no tenga ranura de escritura, si apreciamos que el LED da error y parpadea, puede que las soldaduras no sean correctas y no haya buen contacto; pero si por el contrario observamos que el disco se está formateando con normalidad, quiere decir que el montaje es correcto.

Ahora sólo nos queda instalar el pulsador (o interruptor) en la parte más cómoda para nosotros. Es aconsejable el frontal derecho, a la altura del LED verde de encendido, que se encuentra en la parte izquierda. Realizaremos un taladro de las dimensiones apropiadas al pulsador o interruptor, y procederemos a su colocación, se aconseja previamente haber soldado los cables pues una vez montado no queda mucho espacio para la soldadura.

ADVERTENCIAS: Esta operación invalida la garantía de la unidad de disco, por lo que se aconseja que se efectúe una vez haya concluido ésta.

Si por alguna razón tu unidad no corresponde a la descrita aquí, aconsejamos que desistas de tu empeño y consultes el Manual Técnico de tu unidad.

Si tienes alguna duda, no lo hagas, o díselo a un amigo más entendido. Recuerda las Leyes de **Murphy**, que dicen que si algo puede salir mal, saldrá mal sin duda.

También conviene recordar las que dicen:

- Todo cable cortado a la medida exacta, resultará corto.
- Una herramienta que se nos escape de las manos caerá en el lugar más inaccesible o sobre el componente más frágil.
- Ningún experimento es un fracaso total: como mínimo sirve como un mal ejemplo.

Francisco J. Palacio

NOTA: Tanto el autor como la revista INPUT declinan toda responsabilidad de los daños o desperfectos que se puedan producir a raíz de este montaje.

TABLA 2

PATILLA 01	Negativo para el LED emisor
PATILLA 02	Positivo para el LED emisor
PATILLA 03	No conectada
PATILLA 04	No conectada
PATILLA 05	No conectada
PATILLA 06	No conectada
PATILLA 07	No conectada
PATILLA 08	Positivo para el LED de error
PATILLA 09	Señal negativa para el LED de error
PATILLA 10	No conectada
PATILLA 11	No conectada
PATILLA 12	Entrada de la señal del fotodiodo
PATILLA 13	Negativo para el fotodiodo
PATILLA 14	No conectada
PATILLA 15	No conectada

ADAPTADOR DE DISCO PARA EL COMMODORE

- LOCALIZACION DE LOS SAVE, LOAD, OPEN Y VERIFY
- SOBREESCRITURA DE LOS FICHEROS
- USO DE UNA TABLA DE DATOS
- ADICION DE NUEVOS DATOS

Con el uso de esta sencilla rutina convertidora en código máquina, puedes hacer que los programas de Commodore **que utilizan SAVE y LOAD con cinta funcionen con una unidad de discos.**

En la mayoría de los programas que aparecen en las publicaciones, se ha supuesto que se almacenaban en cinta y se cargaban desde cinta, por medio de las sentencias SAVE y LOAD. Pero cada vez es mayor el número de personas que compran una unidad de disco, la cual resulta mucho más rápida y eficiente en la carga y almacenamiento de programas.

Si tienes unidad de *diskettes*, naturalmente puedes modificar a mano todos los programas. ¿Pero qué te parecería si fuera tu ordenador el que se encargara de hacer la modificación? El programa en lenguaje ensamblador que viene a continuación modificará los programas que trabajen con la unidad de cinta, haciendo que sean utilizables por la de discos en los **Commodore 64** y **Vic 20**.

El primer programa convierte cualquier programa escrito para **Commodore 64**, de forma que funcione con tu unidad de discos.

Lo que hace esencialmente dicho programa es añadir ,8 a continuación de todos los comandos de cinta. El 8 es el número de dispositivo correspondiente a la primera unidad de discos conectada. Pero este programa llega un poco más lejos. Existe una diferencia entre un SAVE ejecutado en cinta o en disco. Al hacer un SAVE en cinta, puedes escribir un programa por encima de otro, o almacenar un fichero o programa tantas veces como quieras bajo el mismo nombre. Sin embargo no se puede almacenar en disco un fichero o programa dos veces con el mismo nombre. Si intentas hacerlo, recibirás un mensaje de error.

Los programas que vayan precedidos de @:[ESPACIO] se escribirán por encima de cualquier versión anterior de los mismos que haya en el disco. Por eso la presente utilidad también inserta @:[ESPACIO] antes del nombre de cualquier programa.

No obstante los nombres de los programas se almacenan a veces en forma de cadenas de caracteres. Para tener esta posibilidad, la utilidad tendría que recorrer todo el programa, encontrar la cadena de caracteres adecuada y añadir @:[ESPACIO] lo cual duplicaría la longitud de la rutina. Por eso no se ha cubierto esta posibilidad. Si alguno de tus programas almacena los nombres de ficheros de esta manera, no tienes más alternativa que añadir tú mismo @:[ESPACIO] entre comillas al principio del nombre del programa.

Programa 1 Teclea para C-64

```
Cargar con ensamblador

*=$C000 ; ORG=49152
        LDA #0
        STA $033E
        CLC
        LDA $2B
        ADC #3
        STA $FB
        LDA $2C
        ADC #$00
        STA $FC
SIGNTE  LDY #0
        INC $FB
        BNE AA
        INC $FC
AA      LDA $FB
        CMP $2D
        BNE BB
        LDA $FC
        CMP $2E
        BNE BB
        JMP STOP
BB      LDA ($FB),Y
        CMP #0
        BNE AB
        INY
        LDA ($FB),Y
        BNE TP
        INY
        LDA ($FB),Y
        BNE TP
        JMP STOP
TP      LDA $FB
        ADC #3
        STA $FB
        LDA $FC
        ADC #0
        STA $FC
AD      LDX #0
        STX $033E
        JMP SIGNTE
AB      CMP #34
        BNE AC
        LDX $033E
        CPX #0
        BNE AD
        LDX #1
        STX $033E
        JMP SIGNTE
AC      LDX $033E
        CPX #1
        BEQ SIGNTE
        CMP #147
        BEQ BUCLE
        CMP #148
        BEQ BUCLE
        CMP #149
        BEQ BUCLE
        CMP #159
        BEQ BUCLE
        JMP SIGNTE
BUCLE   INY
        LDA ($FB),Y
        CMP #32
        BEQ BUCLE
        CMP #0
        BEQ COLINE
        CMP #58
        BEQ COLINE
        CMP #44
```

DATA BECKER

EL Nº 1 EN INFORMATICA

INFORMACION PRODUCTOS *DATA BECKER* FERRE MORET, S.A.

	P.V.P.
* Lenguaje máquina para avanzados	2.332
* Todo sobre bases de datos y gestión de ficheros	2.332
* Todo sobre impresoras CBM 64-128	2.968
* El Commodore 64 como traductor	2.332
* El libro de estadísticas para el Commodore 64	2.968
* 128 Consejos y trucos	2.968
* 128 El libro de primicias	2.332
* 128 El gran libro BASIC	2.544
* 128 Interno	4.240
* 128 Para principiantes	1.908
* 128 Peeks y Pokes	1.908
* CPC 464/6128 Consejos y trucos	2.332
* El manual escolar CPC 464	2.332
* Lenguaje máquina CPC 464/6128	2.332
* El gran libro del floppy CPC	2.968
* CPC Peeks y Pokes 464/6128	1.696
* CPC 6128 Para principiantes	1.908
* MSX El manual escolar	2.968
* MSX Consejos y trucos	2.332
* MSX Gráficos y sonido	2.968
* ZX SPECTRUM consejos y trucos	2.332
* ZX SPECTRUM manual escolar	2.332
* El procesador Z80	4.028
* ATARI ST consejos y trucos	2.968
* PCW 8256 para principiantes	1.908

	P.V.P.
* C-64 En el campo de la ciencia y de la técnica	2.968
* Todo sobre el floppy 1541	3.392
* Lenguaje máquina para Commodore	2.332
* 64 Consejos y trucos	2.968
* Robótica para su Commodore 64	2.968
* Mantenimiento y reparación 1541	2.968
* 64 Interno	4.028
* Peeks y Pokes para Commodore 64	1.696
* Manual del cassette	1.696
* Gráficos para Commodore 64	2.332
* Manual escolar para Commodore 64	2.968
* Consejos y trucos Commodore 64 vol. 2	2.332
* El diccionario para su Commodore 64	2.968
* El ensamblador	2.332
* El libro de ideas del Commodore 64	1.696
* Inteligencia artificial	2.968
* CP/M El libro de ejercicios para CPC	2.968
* CPC consejos y trucos Tomo II	2.544
* MSX Programas y utilidades	2.332
* MSX Lenguaje máquina	2.332
* MSX Para principiantes	1.908
* Metodología de la programación	2.332
* El Commodore 16 para principiantes	1.696
* ATARI ST Peeks y Pokes	1.908

NOVEDAD

Estadística con el C-64. Un libro para la enseñanza y el trabajo que contiene los aspectos básicos de la Estadística con numerosas soluciones y programas BASIC terminados. Todos los que tengan que ver con cálculos estadísticos, sabrán valorar este libro.
El libro de Estadísticas. 448 págs.

SOFTWARE COMMODORE 64-128

PLATINE 64	33.600	Diseñador de circuitos impresos con trazo automático.
ELECTROMAT	4.592	Diseñador de esquemas de circuitos.
PROFIMAT	6.552	Monitor y macroensamblador.
TEXTOMAT	6.552	Procesador de textos con juego de caracteres castellano y catalán.
BASIC 128	6.552	Compilador basic optimizado.
ADA	13.104	Un potente lenguaje de programación.

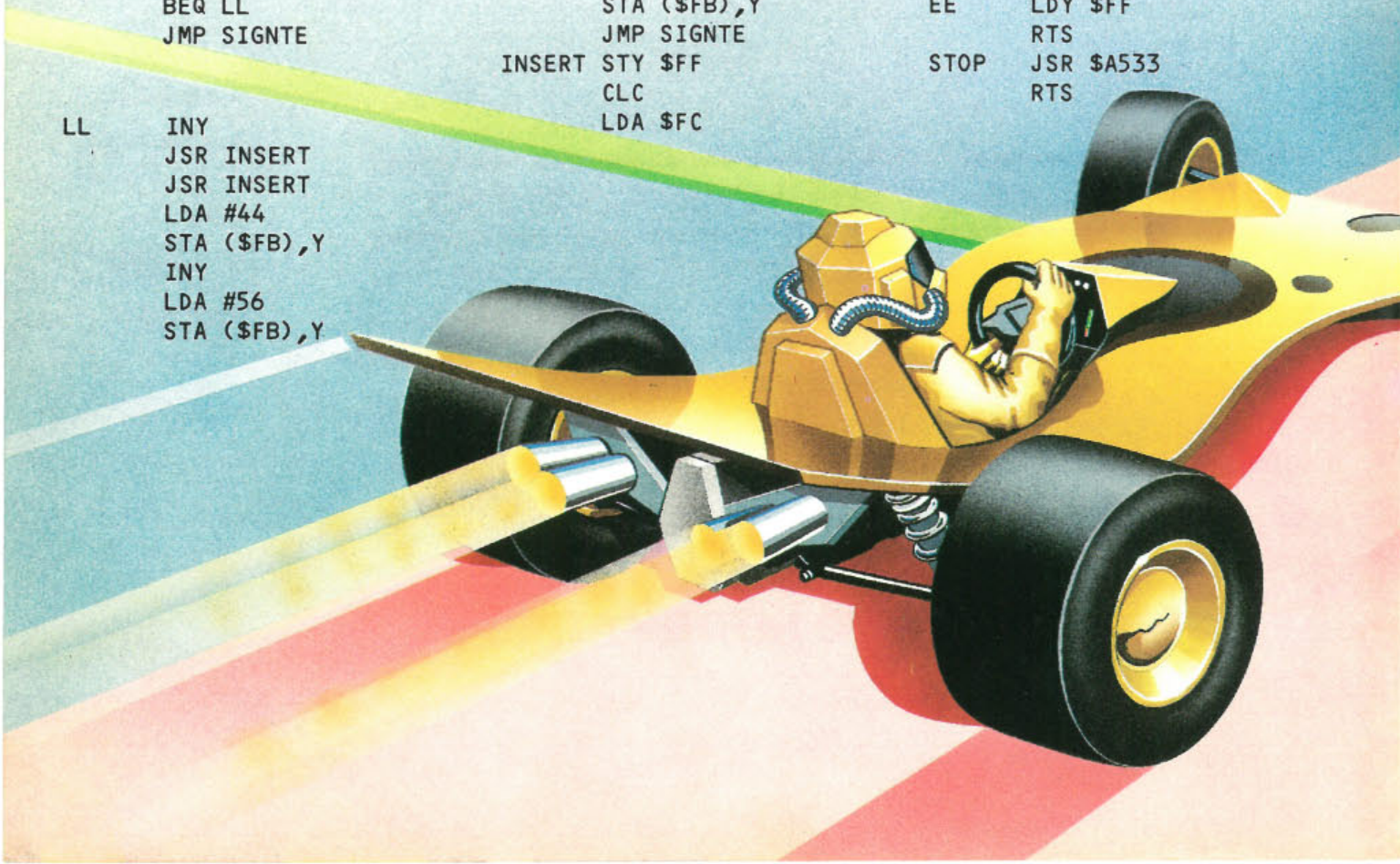
SOLICITE FOLLETO INFORMATIVO

BOLETIN DE PEDIDO
FERRE - MORET S.A.
Córcega, 299
08008 BARCELONA

...o adquirir
...os de envío incluidos.
...BRE
...CCION

I.V.A. Y PORTES INCLUIDOS EN EL PRECIO

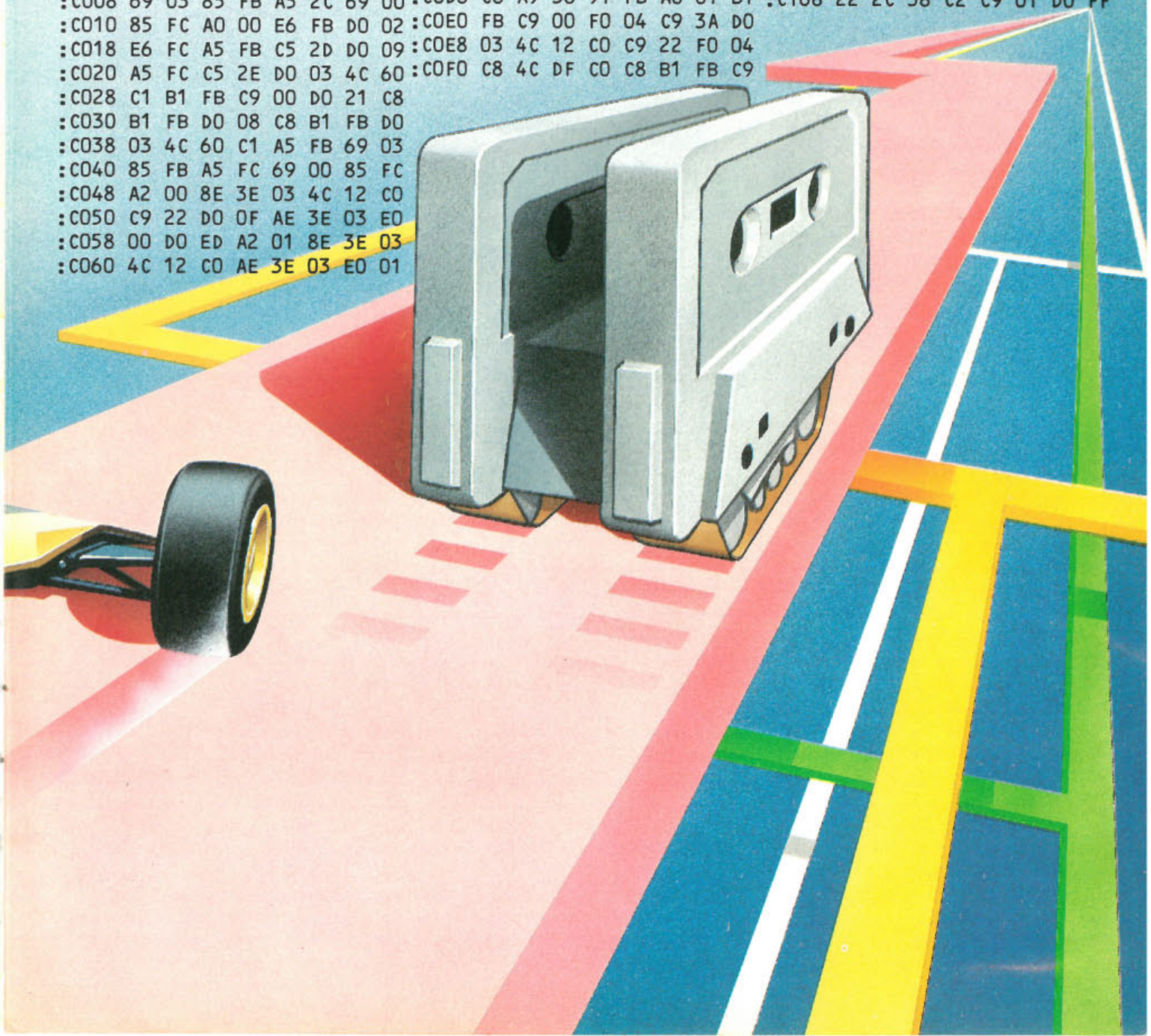
```
        BEQ COMA
        JMP BUCLE
COMA    INY
        LDA ($FB),Y
        CMP #49
        BEQ CG
        JMP SIGNTE
CG      LDA #56
        STA ($FB),Y
        JMP COMILL
COLINE  DEY
        LDA ($FB),Y
        CMP #32
        BEQ COLINE
        CPY #0
        BNE KK
        INY
CC      JSR INSERT
        LDA ETIQ-1,Y
        STA ($FB),Y
        INY
        CPY #8
        BNE CC
        JMP SIGNTE
KK      LDA ($FB),Y
        CMP #34
        BEQ LL
        CMP #36
        BEQ LL
        JMP SIGNTE

LL      INY
        JSR INSERT
        JSR INSERT
        LDA #44
        STA ($FB),Y
        INY
        LDA #56
        STA ($FB),Y

COMILL  LDY #1
RR      LDA ($FB),Y
        CMP #0
        BEQ SS
        CMP #58
        BNE TT
SS      JMP SIGNTE
TT      CMP #34
        BEQ UU
        INY
        JMP RR
UU      INY
        LDA ($FB),Y
        CMP #32
        BEQ UU
        CMP #64
        BNE VV
        JMP SIGNTE
VV      JSR INSERT
        JSR INSERT
        JSR INSERT
        LDA #64
        STA ($FB),Y
        INY
        LDA #58
        STA ($FB),Y
        INY
        LDA #32
        STA ($FB),Y
        JMP SIGNTE
INSERT  STY $FF
        CLC
        LDA $FC
        STA $033D
PP      TYA
        ADC $FB
        BCC QQ
        INC $033D
QQ      STA $033C
        LDA $2D
        STA $FD
        LDA $2E
        STA $FE
SUB     DEC $FD
        LDA #255
        CMP $FD
        BNE DD
        DEC $FE
DD      LDY #0
        LDA ($FD),Y
        LDY #1
        STA ($FD),Y
        LDA $033C
        CMP $FD
        BNE SUB
        LDA $033D
        CMP $FE
        BNE SUB
        INC $2D
        BNE EE
        INC $2E
EE      LDY $FF
        RTS
STOP    JSR $A533
        RTS
```

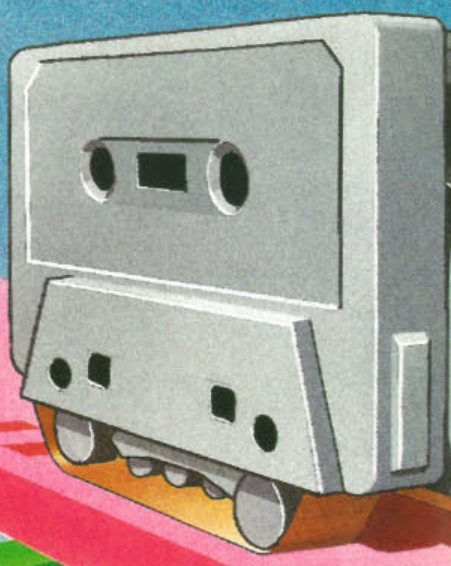


```
ETIQ  .BYTE $22
      .BYTE $40
      .BYTE $3A
      .BYTE $20
      .BYTE $22
      .BYTE $2C
      .BYTE $38
```

Listado para C-64 en hexadecimal, equivalente al anterior en ensamblador.

```
:C000 A9 00 8D 3E 03 18 A5 2B
:C008 69 03 85 FB A5 2C 69 00
:C010 85 FC A0 00 E6 FB D0 02
:C018 E6 FC A5 FB C5 2D D0 09
:C020 A5 FC C5 2E D0 03 4C 60
:C028 C1 B1 FB C9 00 D0 21 C8
:C030 B1 FB D0 08 C8 B1 FB D0
:C038 03 4C 60 C1 A5 FB 69 03
:C040 85 FB A5 FC 69 00 85 FC
:C048 A2 00 8E 3E 03 4C 12 C0
:C050 C9 22 D0 0F AE 3E 03 E0
:C058 00 D0 ED A2 01 8E 3E 03
:C060 4C 12 C0 AE 3E 03 E0 01
:C068 F0 A8 C9 93 F0 0F C9 94
:C070 F0 0B C9 95 F0 07 C9 9F
:C078 F0 03 4C 12 C0 C8 B1 FB
:C080 C9 20 F0 F9 C9 00 F0 1C
:C088 C9 3A F0 18 C9 2C F0 03
:C090 4C 7D C0 C8 B1 FB C9 31
:C098 F0 03 4C 12 C0 A9 38 91
:C0A0 FB 4C DD C0 88 B1 FB C9
:C0A8 20 F0 F9 C0 00 D0 11 C8
:C0B0 20 1C C1 B9 63 C1 91 FB
:C0B8 C8 C0 08 D0 F3 4C 12 C0
:C0C0 B1 FB C9 22 F0 07 C9 24
:C0C8 F0 03 4C 12 C0 C8 20 1C
:C0D0 C1 20 1C C1 A9 2C 91 FB
:C0D8 C8 A9 38 91 FB A0 01 B1
:C0E0 FB C9 00 F0 04 C9 3A D0
:C0E8 03 4C 12 C0 C9 22 F0 04
:C0F0 C8 4C DF C0 C8 B1 FB C9
:C0F8 20 F0 F9 C9 40 D0 03 4C
:C100 12 C0 20 1C C1 20 1C C1
:C108 20 1C C1 A9 40 91 FB C8
:C110 A9 3A 91 FB C8 A9 20 91
:C118 FB 4C 12 C0 84 FF 18 A5
:C120 FC 8D 3D 03 98 65 FB 90
:C128 03 EE 3D 03 8D 3C 03 A5
:C130 2D 85 FD A5 2E 85 FE C6
:C138 FD A9 FF C5 FD D0 02 C6
:C140 FE A0 00 B1 FD A0 01 91
:C148 FD AD 3C 03 C5 FD D0 E7
:C150 AD 3D 03 C5 FE D0 E0 E6
:C158 2D D0 02 E6 2E A4 FF 60
:C160 20 33 A5 60 22 40 3A 20
:C168 22 2C 38 C2 C9 01 D0 FF
```





FUNCIONAMIENTO

Lo primero que hace el programa es cargar con 0 el acumulador y almacenar este valor en la dirección de memoria $033E. Con ello se inicializa un byte, que será utilizado más adelante como indicador (*flag*) en la rutina.

El puntero que señala el principio del BASIC está contenido en las direcciones de memoria 43 y 44 o $2B y $2C en hexadecimal. Las cuatro primeras instrucciones de esta rutina almacenan este puntero en las direcciones $FB Y $FC de la página cero de memoria, que se encuentran en la zona de usuario de la página cero. En dicho lugar tienes acceso a ellas para manipularlas.

A continuación se carga el registro Y con el número 0. La instrucción INC $FB incrementa el contenido del registro $FB; en otras palabras, lo que hace es desplazar el puntero al siguiente byte del programa en BASIC. Si después de realizar el incremento el resultado es cero, no hará nada la siguiente instrucción de BNE AA, que significa saltar si el resultado no es igual a cero, incrementandose el byte alto del puntero. De esta forma se tiene en cuenta automáticamente la transición al llegar al borde de una página.

Seguidamente se carga en el acumulador el contenido del byte bajo del puntero y se compara con el byte bajo del puntero que hay almacenado en las direcciones de memoria 45 y 46 o $2D y $2E en hexadecimal. Este valor señala el final del BASIC. Si ambos bytes bajos coinciden, se pasa a comparar también los bytes altos. Evidentemente, cuando coinciden ambos bytes, el alto y el bajo, es señal de que la rutina ha llegado al final del programa BASIC por lo que salta a STOP, situado al final de la misma.

Caso de que no sea así, el byte al que apuntan $FB y $FC, modificado por el valor del desplazamiento contenido en el registro Y, se carga en el acumulador por medio de la instrucción LDA ($FB),Y. Observa que el valor contenido en el registro Y es siempre 0 en este punto por lo que no hay desplazamiento alguno. Pero se necesita direccionamiento indirecto ya que todas las instrucciones de direccionamiento indirecto son indexadas.

COMPROBACION DE LAS COMILLAS

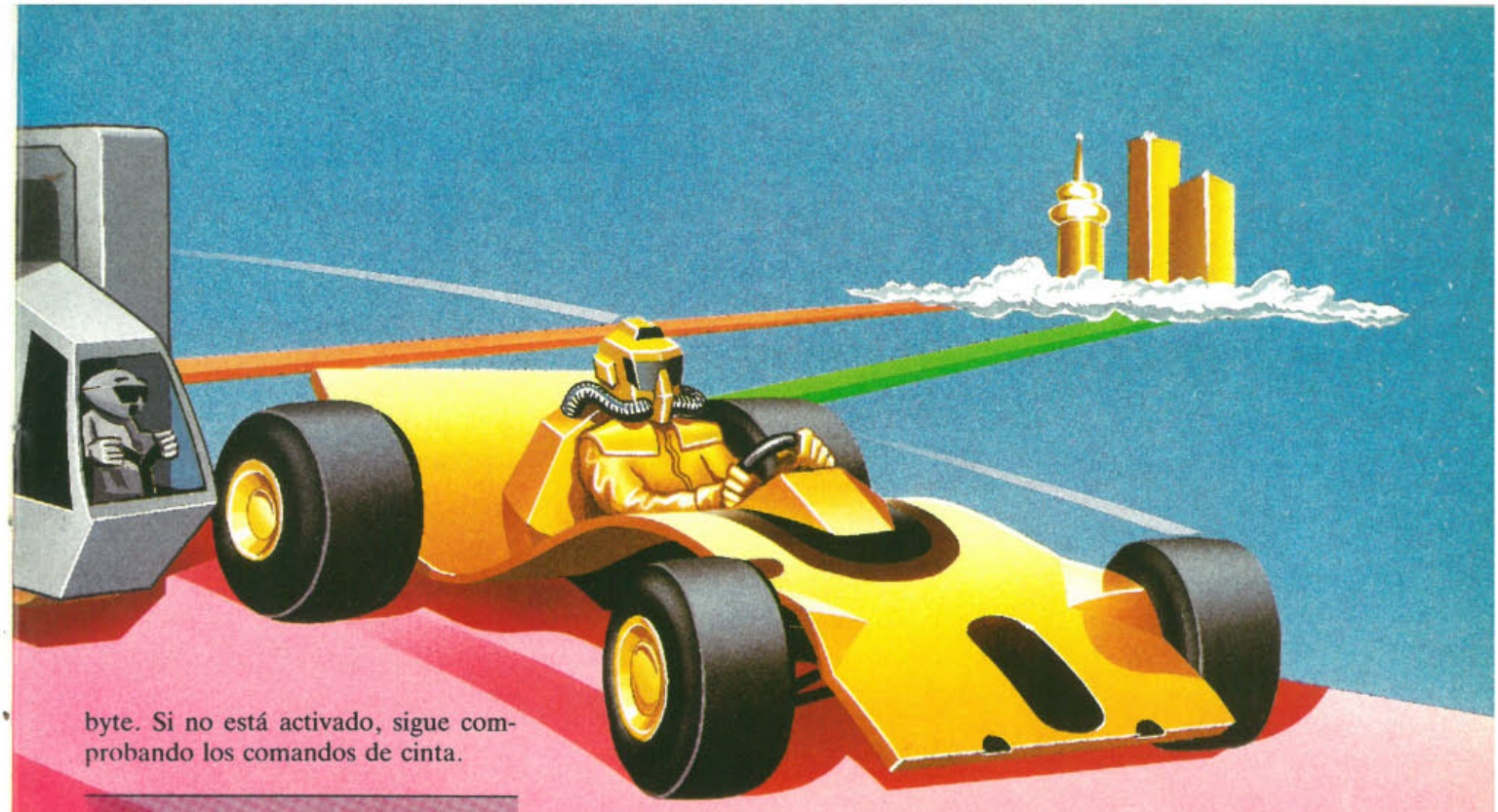
Dos de los símbolos gráficos del teclado del **Commodore** —borrado de pantalla e inserción— pueden confundirse con los comandos de cinta LOAD y SAVE. Los códigos ASCII de los dos símbolos gráficos coinciden con los *tokens* de estos comandos. Pero existe una manera de especificar la diferencia entre ambos comandos. Los símbolos gráficos siempre aparecen entre comillas, mientras que SAVE y LOAD nunca llevan comillas. Incluso aunque aparezcan las palabras SAVE y LOAD entre comillas, no son comandos y no van codificados. Se almacenan como caracteres ASCII por lo que pueden ser presentados en pantalla.

Por eso la rutina tiene que comprobar si el byte que está manejando tiene comillas o no. Esto lo hace activando un indicador que en este caso es la posición de memoria $033E que había sido puesta a cero al principio del programa. Hay que activar dicho *flag* cuando aparecen las primeras comillas en una línea. Cuando aparecen las segundas comillas en esa misma línea hay que volver a poner a cero el indicador. Más adelante en el programa puede examinarse el indicador cuando sea necesario.

La instrucción CMP #0 da comienzo a la rutina de comprobación de las comillas examinando el 0 que va al final del marcador de línea. Esto se hace en el caso de que sólo haya habido unas comillas en una línea. Si se encuentra el valor cero, la instrucción BNE AB no ejecuta el salto, se carga un 0 en el registro X y se almacena en $033E desactivando el indicador. Seguidamente la instrucción JMP NEXT hace que el procesador vuelva al principio, pasando a comprobar el siguiente byte del programa BASIC.

Si no se encuentra un 0, el procesador salta a la etiqueta AB y compara el byte con 34, que es el valor ASCII de las comillas. Si el valor encontrado era 0, la instrucción BNE no actúa. El contenido de la dirección de memoria $033E se carga en el registro X y se compara con 0. Si vale 0, la instrucción BNE AD no ejecuta el salto, se carga el valor 1 en el registro X y se almacena en $033E. Si el valor encontrado no es cero, es decir, si previamente había sido puesto a 1, la instrucción BNE AD hace que el procesador vuelva a la rutina que almacena 0 en $033E para inicializar otra vez dicha dirección. En cualquier caso, se encuentra después con la instrucción JMP NEXT que arranca de nuevo la rutina para que actúe sobre el siguiente byte del BASIC.

Pero si no se encuentran comillas, la instrucción BNE AC lleva el procesador a ejecutar la LDX $033E, que carga en el registro X el contenido de $033E. Este valor se compara con 1 para ver si está activado el indicador o no. Si está activado, la instrucción BEQ NEXT lleva al procesador al comienzo de la rutina para el siguiente



byte. Si no está activado, sigue comprobando los comandos de cinta.

ENCONTRANDO LOS COMANDOS DE CINTA

El siguiente byte del programa en BASIC está todavía en el acumulador y la instrucción CMP lo compara con el número 147, que es el *token* correspondiente a LOAD. Si el valor es precisamente 147, la siguiente instrucción BEQ BUCLE, saltar si el resultado es igual, envía al procesador al lugar donde está situada la etiqueta BUCLE.

Si el valor de dicho byte no es el código correspondiente a LOAD, se pasa a compararlo con 148 que es el de SAVE, después con 149 que es el de VERIFY y por último con 159 que es el de OPEN. Si no se encuentra ninguno de estos comandos de cinta, JMP NEXT envía al procesador de nuevo al principio de la rutina para seguir con el próximo byte. Pero si se encuentra alguno de los códigos anteriores, el procesador se sale de la rutina BUCLE.

LA RUTINA «BUCLE»

La instrucción INY incrementa el registro Y y la LDA ($FB),Y carga en el acumulador el siguiente byte del comando de cinta. A continuación la rutina utiliza CMP #32 para comprobar si hay un espacio. 32 es el código ASCII del espacio.

Si el siguiente carácter es un espacio, la instrucción BEQ BUCLE envía al procesador de nuevo al principio de la rutina BUCLE para ver si el carácter siguiente es también un espacio.

En caso contrario, la instrucción CMP #0 comprueba si hay un cero en la marca de final de línea. Si se encuentra un 0, la instrucción BEQ COLINEA envía al procesador a la rutina COLINEA.

Lo siguiente que se comprueba es la existencia de una coma. La instrucción CMP #44 compara el valor del byte con el código ASCII correspondiente a la coma (44). Si el siguiente byte es una coma, la instrucción BEQ COMA envía al procesador a ejecutar la rutina COMA. De no ocurrir esto, la instrucción JMP LOOP ejecuta un salto hacia atrás al comienzo de la rutina BUCLE, de forma que pueda comprobarse en el siguiente byte la existencia de un espacio, final de línea, dos puntos o coma.

LA RUTINA «COMA»

Si en la rutina BUCLE se encuentra una coma, el procesador pasa a ejecutar la rutina COMA. Lo primero que esta hace es incrementar el contenido del registro Y; posteriormente la instrucción LDA ($FB),Y carga el acumulador con el contenido del byte que sigue a la coma.

Este valor se compara con 49, que es el código ASCII del número 1. El número 1 colocado a continuación de un comando de salida significa que se va a usar el dispositivo número 1, que es el magnetófono. El número 2 se utilizaría para designar a la pantalla, el 4 una impresora y el 8 una unidad de discos. Por eso si lo que sigue no es un 1, la rutina salta de nuevo al principio.

Si el valor del byte siguiente es el código ASCII del número 1, la instrucción LDA #56 carga el acumulador con el valor 56, que es el código ASCII de 8. Este valor se almacena en la dirección adecuada de memoria por medio de la instrucción STA ($FB),Y. El procesador salta entonces a la rutina COMILLAS.

LA RUTINA «COLINEA»

Lo primero que hace la rutina COLINEA es decrementar el registro Y. Seguidamente carga el acumulador con el contenido del byte que hay inmediatamente antes de los dos puntos o del marcador de final de línea. Si se trata de un espacio, las instrucciones CMP #32 y BEQ COLINEA fuerzan

un retorno al principio. En otras palabras, la rutina se mueve hacia atrás desde los dos puntos o el marcador de final de línea hasta que se encuentra con un carácter que no es un espacio. Lo que va buscando es un nombre de programa colocado entre comillas.

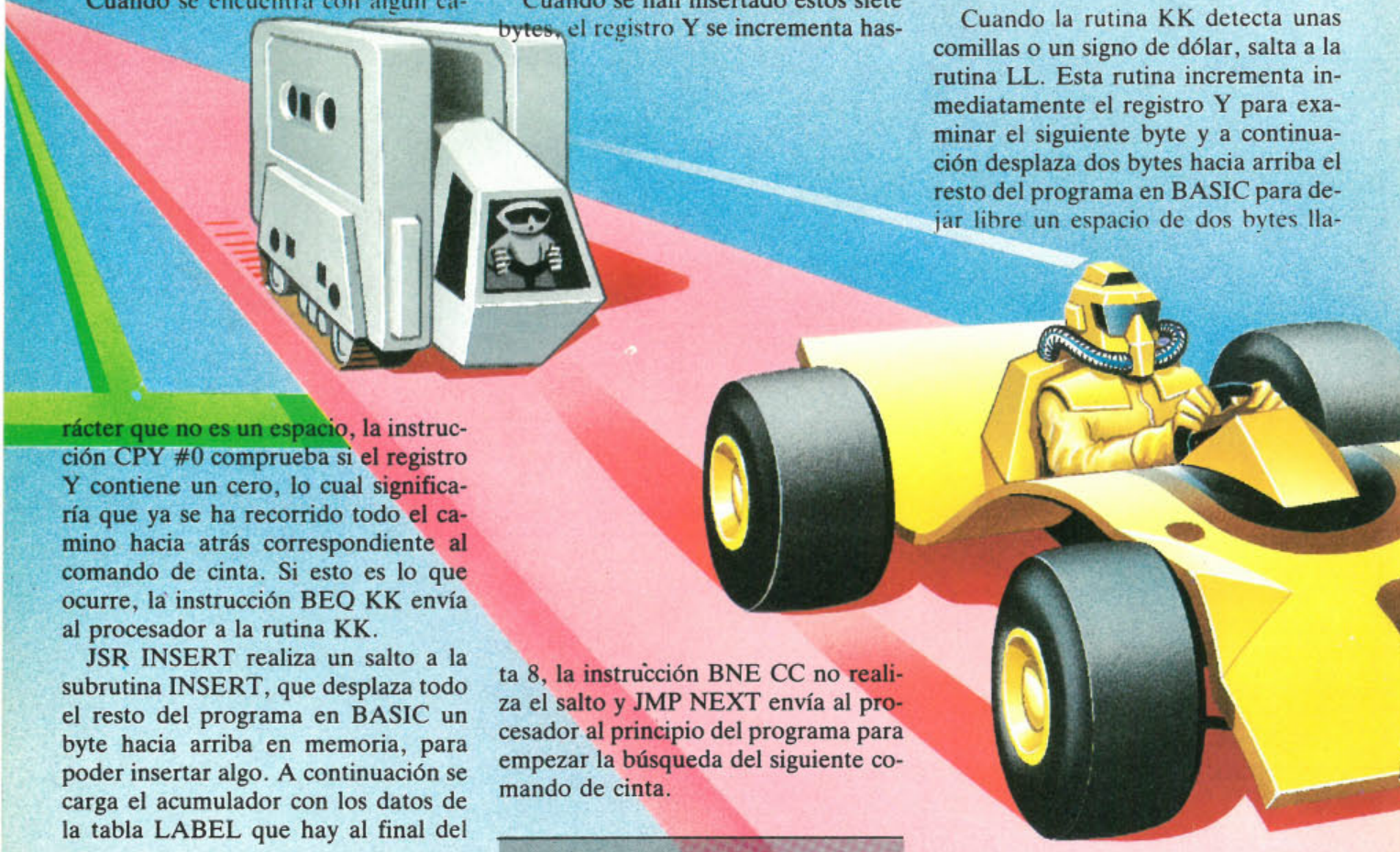
Cuando se encuentra con algún carácter que no es un espacio, la instrucción CPY #0 comprueba si el registro Y contiene un cero, lo cual significaría que ya se ha recorrido todo el camino hacia atrás correspondiente al comando de cinta. Si esto es lo que ocurre, la instrucción BEQ KK envía al procesador a la rutina KK.

JSR INSERT realiza un salto a la subrutina INSERT, que desplaza todo el resto del programa en BASIC un byte hacia arriba en memoria, para poder insertar algo. A continuación se carga el acumulador con los datos de la tabla LABEL que hay al final del programa.

Recuerda que esta parte de programa sólo se ejecuta cuando no hay nada a continuación del comando de cinta, a excepción de los espacios, los dos puntos y las marcas de fin de línea. De esta forma Y empieza conteniendo el valor 0. Posteriormente se incrementa para apuntar al primer byte que sigue al comando y la instrucción LDA LABEL - 1,Y carga el acumulador con el primer byte de la tabla LABEL. Este byte se almacena en el programa BASIC. INY incrementa el registro Y, CPY compara con 8 el contenido de dicho registro y, si todavía no ha llegado a 8, BNE CC envía al procesador a desplazar nuevamente hacia arriba el programa en BASIC y a almacenar otro byte de la tabla.

Los siete bytes de datos de la tabla LABEL son 22, 40, 3A, 20, 22, 2C y 38 que son los códigos ASCII hexadecimales de los caracteres « @ : [ESPACIO] » , 8. Con esto el sencillo comando LOAD de cinta se convierte en LOAD «@:[ESPACIO]»,8.

Cuando se han insertado estos siete bytes, el registro Y se incrementa hasta 8, la instrucción BNE CC no realiza el salto y JMP NEXT envía al procesador al principio del programa para empezar la búsqueda del siguiente comando de cinta.



LA RUTINA KK

Si la rutina que estaba comprobando los dos puntos o la marca de final de línea se encuentra con un carácter diferente al código del correspondiente comando de almacenamiento en cinta, se llama a la rutina KK que comprueba las comillas o los signos de dólar.

Lo primero que hace esta rutina es cargar el byte en cuestión. CMP #34, BEQ LL y CMP #36, BEQ LL comprueban la existencia de comillas o el signo de dólar y envían al procesador a LL cuando se presenta uno de ambos signos. Si el byte contiene un carácter que no es ninguno de éstos, se trata de un carácter ilegal en las presentes circunstancias. En consecuencia la rutina lo ignora y salta a la etiqueta NEXT para comprobar el siguiente carácter.

LA RUTINA LL

Cuando la rutina KK detecta unas comillas o un signo de dólar, salta a la rutina LL. Esta rutina incrementa inmediatamente el registro Y para examinar el siguiente byte y a continuación desplaza dos bytes hacia arriba el resto del programa en BASIC para dejar libre un espacio de dos bytes llamando dos veces a la subrutina INSERT.

En esos dos espacios vacíos hay que cargar ,8. Con ello el comando SAVE «nombre», SAVE P$ o como quiera que hayas llamado a tu programa, se transforma en el comando de disco SAVE «nombre»,8 o SAVE P$,8.

Así, el acumulador se carga con 44,

que es el código ASCII de la coma y este valor se almacena en la posición adecuada de memoria por medio de la instrucción STA ($FB),Y. Seguidamente se incrementa el registro Y y el acumulador se carga con 56, que es el código ASCII del 8, por lo que la misma instrucción STA ($FB),Y carga el 8 en el siguiente byte.

LA RUTINA «COMILLAS»

La rutina COMILLAS empieza poniendo a 1 el valor de Y, por medio de la instrucción LDY #1 y seguidamente carga el primer byte que sigue al código de la instrucción de cinta por medio de LDA ($FB),Y. Este valor se compara entonces con 0 y 58 para comprobar cuándo se alcanza el final de la sentencia BASIC en cuestión.

Se utiliza aquí una construcción bastante peculiar por medio de BEQ SS que salta a la instrucción de salto incondicional JMP NEXT. Se hace así porque la instrucción de salto condicional no permite un salto de amplitud superior a 128 bytes hacia atrás o 127 bytes hacia adelante. Las instrucciones de salto condicional sólo llevan un byte para el valor del salto, mientras que las de salto incondicional permiten que la amplitud de dicho salto se exprese por medio de dos bytes.

La siguiente prueba que se realiza es la de las comillas; cuando la instrucción CMP #34 encuentra unas, BEQ UU realiza un salto a la rutina UU. Si no se encuentran comillas, se incrementa el contenido de Y y JMP RR realiza un salto para pasar a comprobar el próximo byte. De esta forma el procesador recorre este bucle una vez y otra hasta que se encuentra con una marca de final de línea, tal como dos puntos o unas comillas.

Cualquiera que sea la forma en que se incremente el registro Y, la prueba CMP #34 sólo permite que INY se utilice una vez antes de cada salto condicional en lugar de las dos veces posteriores a cada salto, con lo cual se ahorra un byte. La instrucción INY afecta al indicador de cero que utiliza BEQ UU para decidir en qué forma ha de saltar.

A continuación se compara el siguiente byte con 32, que es el código ASCII correspondiente al espacio. Si hay un espacio, la rutina inicia el bucle y el registro Y se incrementa nuevamente hasta que en la comprobación se llega a algo diferente de un espacio.

El siguiente carácter que se comprueba es la @. Evidentemente, si el nombre del programa ya tiene una @:[ESPACIO] para permitir que se escriba sobre él, no tienes que añadirle nada más. Por eso cuando el siguiente byte es 64 —el código ASCII de la @— el procesador salta a NEXT. En caso contrario se desplaza a VV y llama tres veces a la rutina INSERT, desplazando todo el resto del programa BASIC tres bytes hacia arriba en la memoria.

En el espacio de tres bytes así creado se almacenan ahora los bytes 64, 58 y 32. Estos tres números son los códigos ASCII de @ : y espacio respectivamente. Una vez que ha sido hecho esto, la instrucción JMP NEXT envía al procesador de nuevo a comprobar el resto del programa en BASIC.

LA RUTINA «INSERT»

La rutina INSERT ya ha sido llamada varias veces durante la ejecución del programa en código máquina. Su efecto es desplazar todo el programa en BASIC un byte hacia arriba en la memoria a partir del punto alcanzado.

Lo primero que hace dicha rutina es almacenar el contenido del registro Y en la dirección de memoria FF de la página cero. El contenido del registro Y ha de ser después de salir de la rutina INSERT el mismo que había al entrar a la rutina, ya que de no ser así perderías tu posición en el programa. Pero durante la ejecución de la rutina INSERT vas a necesitar el registro Y. Por eso se almacena su contenido al principio y, como habrás observado, el contenido de FF se carga nuevamente al final en el registro Y.

La instrucción CLC pone entonces a cero el indicador de acarreo, dejándolo listo para las sumas que vengan después. El valor del byte alto del puntero, almacenado en la dirección $FC, se copia entonces en la dirección de memoria $033D por medio de las instrucciones LDA $FC y STA $033D.

El contenido del registro Y, que es el que lleva la cuenta de cuántos bytes ha pasado el programa a partir de la instruccción de cinta, se transfiere al acumulador de modo que pueda sumarse al byte bajo del puntero conte-

nido en $FB y $FC, que corresponde a las direcciones del propio código de la instrucción de cinta. De esta forma el resultado de la suma da la dirección del último byte del programa en BASIC que no será desplazado dentro de la memoria.

El resultado se almacena en $033C tanto si se desbordan los ocho bits del acumulador como si no ocurre así. Pero cuando ocurre, queda activado el indicador de acarreo, la instrucción BCC —saltar si el acarreo es cero— no realiza el salto condicional y INC $033D incrementa el byte alto del puntero.

La variable del sistema que está apuntando al principio de la zona de variables, es decir, un byte posterior al final del área de BASIC, está contenida en $2D y $2E. Dicho valor se copia en $FD y $FE de forma que puede ser utilizado.

A continuación de la etiqueta SUB hay una pequeña rutina que decrementa el puntero contenido en $FD y $FE. La primera vez que esto ocurre el puntero se decrementa y se queda apuntando al último byte del programa en BASIC. Pero esta pequeña rutina será utilizada una vez y otra para desplazar hacia abajo el programa BASIC byte a byte.

El registro Y se carga con el valor 0 y el acumulador se carga con el contenido de la dirección de memoria a la que apunta el contenido de $FD y $FE, aumentada en el desplazamiento indicado por Y. Y se incrementa en 1 y la instrucción STA ($FD),Y almacena de nuevo el byte una posición de memoria más arriba.

El puntero que estaba almacenado en $033C y $033D, que era el que te decía en qué punto del programa BASIC te encontrabas, se compara con el valor contenido en $FD y $FE. Si ambos valores no coinciden y la rutina no ha completado su trabajo en toda la extensión que debe abarcar la inserción, se produce un salto condicional a la etiqueta SUB, en la que se decrementan los punteros y se desplaza el siguiente byte.

Cuando coinciden los bytes de ambos punteros, significa que ya ha sido desplazado todo el programa BASIC que sigue a la inserción. Lo único que le queda por hacer a la rutina es actualizar el final de la zona de variables BASIC del sistema. Esto se hace con las instrucciones INC $2D, BNE EE e INC $2E.

La instrucción LDY $FF restaura el valor del registro Y y RTS realiza el retorno a la instrucción desde la que se llamó a la rutina.

PONIENDOLO TODO EN ORDEN

Al desplazar el programa BASIC en la memoria, han quedado afectados todos los punteros que apuntan al principio de cada línea del BASIC. Dichos punteros se almacenan al principio de la línea anterior del BASIC.

Afortunadamente existe una rutina

PRECIOS INCLUIDO I.V.A.

MICRO-1

SOMOS MAYORISTAS

C/. DUQUE DE SESTO, 50. 28009 MADRID
METRO O'DONNELL O GOYA

SOFTWARE: ¡¡2 PROGRAMAS POR EL PRECIO DE 1!!
Y ADEMAS, COMPLETAMENTE GRATIS UNA CALCULADORA. INCREIBLE ¿VERDAD?

LEADERBOARD	2.600 ptas.	KNIGHT GAMES	2.300 ptas.
MAP GAME	2.300 ptas.	NEXUS	2.300 ptas.
GREEN BERET	2.300 ptas.	«V»	2.300 ptas.
ROCK'N LUCHA	2.500 ptas.	YIE AR KUNG FU	2.300 ptas.
THE WAY OF THE TIGER	2.500 ptas.	SABOTEUR	2.300 ptas.
GOONIES	2.600 ptas.	CAULDRON II	2.300 ptas.
RAMBO	2.300 ptas.	ZORRO	2.100 ptas.
PING PONG	2.300 ptas.	WORLD CUP	2.500 ptas.

SOFTWARE DE REGALO (OFERTA 2 × 1):
FIGHTING WARRIOR - DUMMY RUN - INDIANA JONES - POLE POSITION

¡¡20% DE DTO. SOBRE P.V.P. EN IMPRESORAS!!

RATON PARA COMMODORE	9.900 ptas.
TAPA COMMODORE 64	1.100 ptas.
JOYSTICK QUICK SHOT I	1.395 ptas.
JOYSTICK QUICK SHOT II	1.695 ptas.
JOYSTICK QUICK SHOT V	1.695 ptas.

Pedidos contra reembolso sin ningún gasto de envío. Teléfs. (91) 275 96 16 - 274 75 02,
o escribiendo a: MICRO-1. C/. Duque de Sesto, 50. 28009 Madrid

INPUT commodore

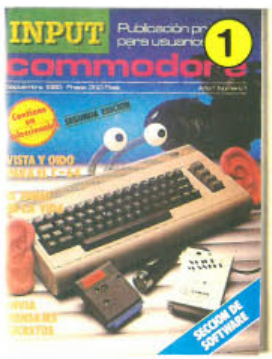
SERVICIO DE EJEMPLARES ATRASADOS

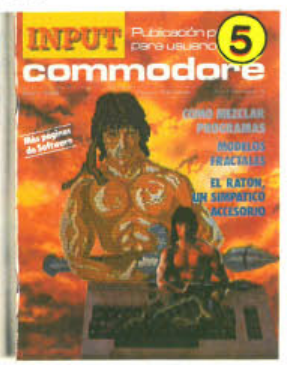
¡NO TE PIERDAS NI UN SOLO EJEMPLAR!

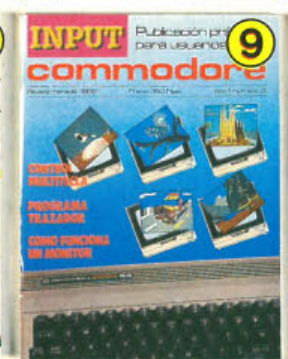
INPUT COMMODORE quiere proporcionar a sus lectores este nuevo servicio de ejemplares atrasados para que no pierdan la oportunidad de tener en sus hogares todos los ejemplares de esta revista, líder en el mercado español.

Podréis solicitar cualquier número de **INPUT COMMODORE** que querais, siempre al precio de cubierta (sin más gastos).

Utiliza el cupón adjunto, enviándolo a **EDISA** (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid, o bien llámanos por teléfono al (91) 415 97 12.







INPUT commodore

CUPON DE PEDIDO

SI, envíenme contrarreembolso ..... ejemplares de **INPUT COMMODORE** de los números:

(marca con una (X) tu elección)

1	2	3	4	5	6	7	8	9	10	11
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

NOMBRE

APELLIDOS

DOMICILIO

NUM. PISO ESCALERA COD. POSTAL

POBLACION PROV.

TELEFONO FIRMA

en ROM que los corrige a todos ellos. Por ello no tienes más que llamar a la subrutina que hay en $A533, por medio de la instrucción JSR $A533. La instrucción RTS se encarga de que el ordenador vuelva al BASIC.

Lo que sigue no es exactamente lenguaje ensamblador. Son datos. El BYTE es una directiva del ensamblador que le ordena que disponga un byte de la memoria para los datos que siguen.

Para llamar al programa, utiliza:

SYS 49152

No te olvides sin embargo de introducir un programa en BASIC, que es el que necesita la conversión, ya que en caso contrario la llamada a la rutina en código máquina será un desperdicio de tiempo, y además si te olvidas se romperá la rutina.

Al hacer la carga del código máquina has de añadir en la instrucción LOAD ,8,1 con lo que ahora quedará así:

LOAD «Nombre del programa»,8,1

Con ,8 se informa a tu ordenador que se está utilizando una unidad de discos, mientras que ,1 ordena a la máquina que ponga el código en el mismo lugar de la memoria de donde se tomó. Teclea ahora NEW para resetear lor punteros del BASIC. Ya puedes cargar el programa a convenir.

VERSION PARA VIC-20

El programa en lenguaje ensamblador que acabamos de ver funcionará también en el **Vic 20**. Pero tienes que reubicarlo, ya que lo que sigue a 49152 no es un área protegida. Ensaya la utilización de 7168 como origen, pero no te olvides de modificar los punteros para desplazar RAMTOP antes de protegerla. Para hacerlo, utiliza los siguientes POKEs:

Teclea para Vic-20

```
POKE 51,255
POKE 52,27
POKE 55,255
POKE 56,27
```

Y después CLR

También es diferente la ROM que sitúa los punteros de línea del BASIC inmediatamente después de todos los desplazamientos que se hacen al final de la rutina. Empieza en $C533 de forma que la instrucción será JSR $C533.

Si no dispones de un ensamblador comercial para tu **Vic 20**, puedes introducir el programa con ayuda del monitor en código máquina (página 280). Si el ensamblado a mano no te asusta, aquí tienes el código máquina que puedes introducir directamente:

Listado para Vic-20 en hexadecimal.

```
A9 00 8D 3E 03 18 A5 2B
69 03 85 FB A5 2C 69 00
85 FC A0 00 E6 FB D0 02
E6 FC A5 FB C5 2D D0 09
A5 FC C5 2E D0 03 4C 60
1D B1 FB C9 00 D0 21 C8
B1 FB D0 08 C8 B1 FB D0
03 4C 60 1D A5 FB 69 03
85 FB A5 FC 69 00 85 FC
A2 00 8E 3E 03 4C 12 1C
C9 22 D0 0F AE 3E 03 E0
00 D0 ED A2 01 8E 3E 03
4C 12 1C AE 3E 03 E0 01
F0 A8 C9 93 F0 0F C9 94
F0 0B C9 95 F0 07 C9 9F
F0 03 4C 12 1C C8 B1 FB
C9 20 F0 F9 C9 00 F0 1C
C9 3A F0 18 C9 2C F0 03
4C 7D 1C C8 B1 FB C9 31
F0 03 4C 12 1C A9 38 91
FB 4C DD 1C 88 B1 FB C9
20 F0 F9 C0 00 D0 11 C8
20 1C 1D B9 63 1D 91 FB
C8 C0 08 D0 F3 4C 12 1C
B1 FB C9 22 F0 07 C9 24
F0 03 4C 12 1C C8 20 1C
1D 20 1C 1D A9 2C 91 FB
C8 A9 38 91 FB A0 01 B1
FB C9 00 F0 04 C9 3A D0
03 4C 12 1C C9 22 F0 04
C8 4C DF 1C C8 B1 FB C9
20 F0 F9 C9 40 D0 03 4C
12 1C 20 1C 1D 20 1C 1D
20 1C 1D A9 40 91 FB C8
A9 3A 91 FB C8 A9 20 91
FB 4C 12 1C 84 FF 18 A5
FC 8D 3D 03 98 65 FB 90
03 EE 3D 03 8D 3C 03 A5
2D 85 FD A5 2E 85 FE C6
FD A9 FF C5 FD D0 02 C6
FE A0 00 B1 FD A0 01 91
FD AD 3C 03 C5 FD D0 E7
AD 3D 03 C5 FE D0 E0 E6
2D D0 02 E6 2E A4 FF 60
20 33 A5 60 22 40 3A 20
22 2C 38 C2 C9 01 D0 FF
```

Para llamar al programa teclea SYS 7168.

No te olvides sin embargo de introducir primero el programa de BASIC que necesita ser convertido, ya que en caso contrario la llamada a la rutina en código máquina no será más que una pérdida de tiempo y además dicha rutina se romperá.

Para almacenar este programa tienes que utilizar tu monitor en código máquina.

Cuando cargues el código máquina desde cinta debes añadir a la instrucción LOAD ,8,1 con lo que quedará así:

LOAD «Nombre del programa»,8,1

Los signos ,8 indican al ordenador que se está utilizando una unidad de disco, mientras que ,1 ordena a la máquina que ponga el código en el mismo lugar de la memoria del que se tomó. Seguidamente teclea NEW para resetear los punteros del BASIC.

Si se te hace difícil encontrar INPUT en tu kiosco habitual, resérvalo por adelantado, o háznoslo saber para que podamos remediarlo

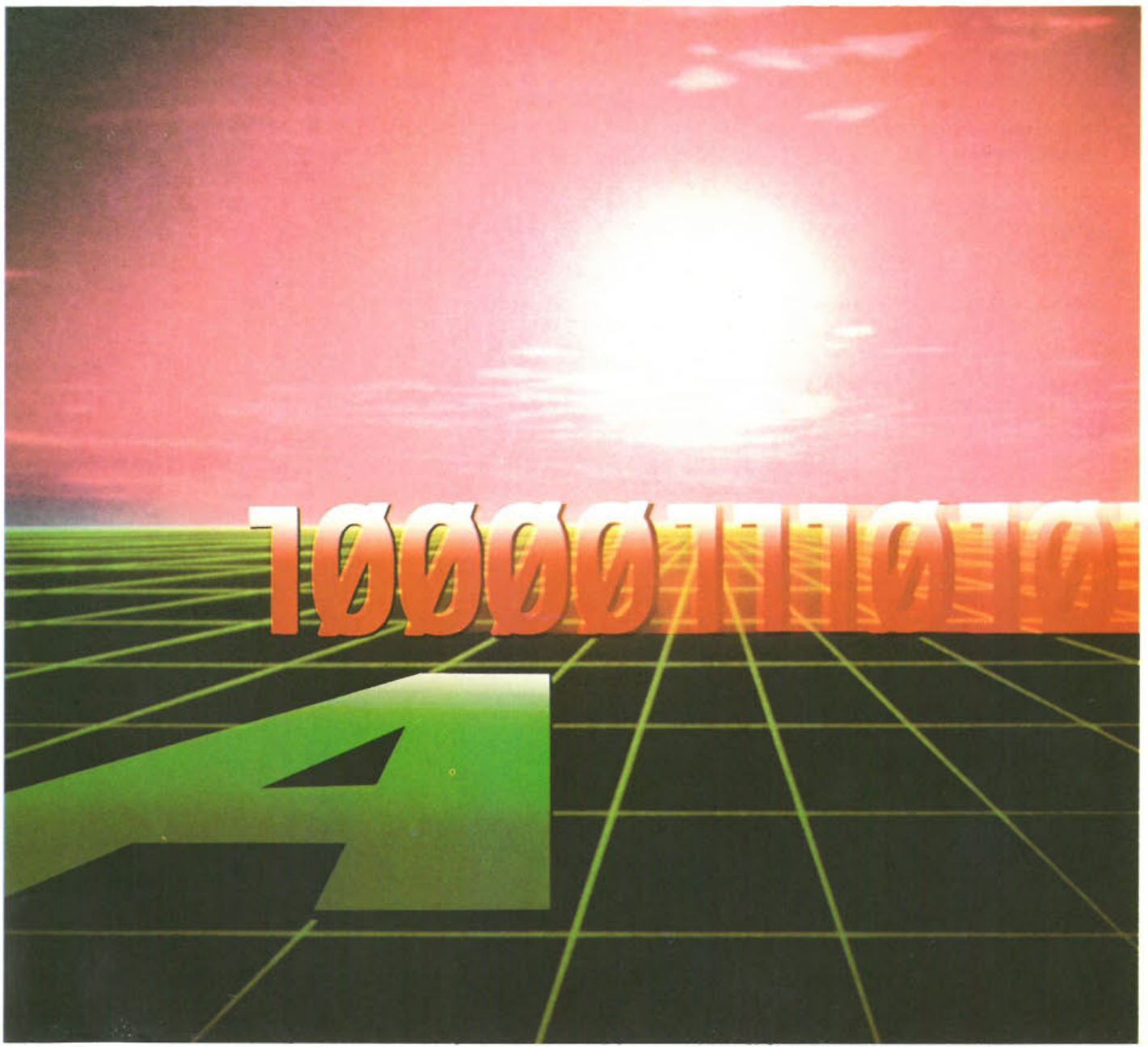
DECODIFICACION DE TEXTOS

- LISTADO DEL PROGRAMA DECODIFICADOR
- COMO SE PRODUCE LA DECODIFICACION
- CONVERSION DEL MATERIAL CODIFICADO
- RECONVERSION DE LOS CODIGOS A BASIC

Completamos el compresor de texto añadiendo la rutina de decodificación. Por si no dispones de un ensamblador adecuado, aquí tienes también un listado completo en hexadecimal de todo el programa compresor.

En el capítulo anterior vimos la manera de codificar textos en binario de forma que ocupen menos cantidad de memoria que con la codificación habitual en ASCII. Los listados en lenguaje ensamblador que le acompañan corresponden a un programa que realiza esta función.

Sin embargo, el limitarse únicamente a la codificación es algo que sirve de bien poco; lo único que conseguirías con ello sería encontrarte frente a una incomprensible masa de número binarios. Por eso hace falta un programa que sea capaz de realizar la decodificación. Los listados que siguen a continuación permitirán añadir al programa la sección decodificadora que te convierta de nuevo el mensaje en texto legible. También veremos la versión hexadecimal completa del compresor de textos, que será de utilidad en el caso de que no poseas un ensam-



blador adecuado. Una vez que hayas completado el programa, almacénalo en cinta.

En el próximo capítulo veremos cómo utilizar el programa en un juego de aventuras, así como la forma de desarrollar juegos con el compresor de textos.

FUNCIONAMIENTO

Partiendo desde el principio del texto codificado en binario, la máquina toma los bits de uno en uno, literalmente construyendo un número binario cada vez más largo. Después de la adición de cada nuevo bit, el número se compara con los códigos de la tabla de conversión, buscando algún emparejamiento. Si no se encuentra ninguno o, por el contrario, se localiza más de uno, el programa añade al número un nuevo bit. El procedimiento se repite hasta que se encuentra un emparejamiento exclusivo para el código binario.

Si la lectura del código se inicia en el punto adecuado, inevitablemente se ha de encontrar un emparejamiento único.

Después de esto, el carácter se inserta en una cadena, dejándolo listo para ser utilizado en BASIC. El código binario existente se elimina y se pasa a construir el siguiente número. El ordenador se entera de que el trozo de texto está completo cuando se encuentra con el código de «fin de mensaje». El programa en BASIC, del que nos ocuparemos en el siguiente capítulo de esta serie, se encargará entonces de visualizar el texto sobre la pantalla.

Si has decidido recurrir al listado en lenguaje ensamblador, tienes que llamar primero al código fuente que almacenaste la vez anterior. A continuación se ensambla todo el conjunto siguiendo las instrucciones del ensamblador que utilices. Evidentemente, también tienes que conservar el código máquina resultante en cinta.

Teclea para C-64 y Vic-20

```
,CE7D 20 41 53 JSR $5341
,CE80 4F       .BYTE $4F
,CE81 54       .BYTE $54
,CE82 52       .BYTE $52
,CE83 49 4C    EOR #$4C
,CE85 45 43    EOR $43
,CE87 5E 5F 55 LSR $555F,X
,CE8A 4D 50 57 EOR $5750
,CE8D 59 4E 42 EOR $424E,Y
,CE90 47       .BYTE $47
,CE91 44       .BYTE $44
,CE92 46 56    LSR $56
,CE94 48       PHA
,CE95 4B       .BYTE $4B
,CE96 51 58    EOR ($58),Y
,CE98 4A       LSR
,CE99 5A       .BYTE $5A
,CE9A 5B       .BYTE $5B
,CE9B 5C       .BYTE $5C
,CE9C 5D 0A 11 EOR $110A,X
,CE9F 04       .BYTE $04
,CEA0 11 17    ORA ($17),Y
,CEA2 08       PHP
,CEA3 11 08    ORA ($08),Y
,CEA5 05 17    ORA $17
,CEA7 02       .BYTE $02
,CEA8 07       .BYTE $07
,CEA9 11 08    ORA ($08),Y
,CEAB 03       .BYTE $03
,CEAC 01 01    ORA ($01,X)
,CEAE 14       .BYTE $14
,CEAF 08       PHP
,CEB0 08       PHP
,CEB1 08       PHP
,CEB2 04       .BYTE $04
,CEB3 08       PHP
,CEB4 08       PHP
,CEB5 08       PHP
,CEB6 0C       .BYTE $0C
,CEB7 0E 0C 08 ASL $080C
,CEBA 0B       .BYTE $0B
,CEBB 0B       .BYTE $0B
,CEBC 00       BRK
,CEBD 0B       .BYTE $0B
,CEBE 04       .BYTE $04
,CEBF 04       .BYTE $04
,CEC0 08       PHP
,CEC1 15 06    ORA $06,X
,CEC3 03       .BYTE $03
,CEC4 04       .BYTE $04
,CEC5 06 14    ASL $14
,CEC7 03       .BYTE $03
,CEC8 04       .BYTE $04
,CEC9 11 02    ORA ($02),Y
,CECB 01 08    ORA ($08,X)
,CECD 17       .BYTE $17
,CECE 03       .BYTE $03
,CECF 04       .BYTE $04
,CED0 07       .BYTE $07
,CED1 17       .BYTE $17
,CED2 06 03    ASL $03
,CED4 06 01    ASL $01
,CED6 06 1A    ASL $1A
,CED8 04       .BYTE $04
,CED9 03       .BYTE $03
,CEDA 06 0A    ASL $0A
,CEDC 1E 0B 0A ASL $0A0B,X
,CEDF 00       BRK
,CEE0 01 03    ORA ($03,X)
,CEE2 07       .BYTE $07
,CEE3 0F       .BYTE $0F
,CEE4 1F       .BYTE $1F
,CEE5 3F       .BYTE $3F
,CEE6 7F       .BYTE $7F
,CEE7 FF       .BYTE $FF
,CEE8 FE FC F8 INC $F8FC,X
,CEEB F0 E0    BEQ $CECD
,CEED C0 80    CPY #$80
,CEEF 00       BRK
,CEF0 48       PHA
,CEF1 98       TYA
,CEF2 48       PHA
,CEF3 A0 00    LDY #$00
,CEF5 B1 2D    LDA ($2D),Y
,CEF7 C8       INY
,CEF8 C9 5A    CMP #$5A
,CEFA D0 06    BNE $cf02
,CEFC B1 2D    LDA ($2D),Y
,CEFE C9 80    CMP #$80
,CF00 F0 06    BEQ $cf08
,CF02 98       TYA
,CF03 69 06    ADC #$06
,CF05 A8       TAY
,CF06 D0 ED    BNE $ceF5
,CF08 C8       INY
,CF09 8C 48 CF STY $cf48
,CF0C 68       PLA
,CF0D A8       TAY
,CF0E 68       PLA
,CF0F 60       RTS
,CF10 48       PHA
,CF11 8A       TXA
,CF12 48       PHA
,CF13 98       TYA
,CF14 48       PHA
,CF15 A5 2F    LDA $2F
,CF17 85 61    STA $61
,CF19 A5 30    LDA $30
,CF1B 85 62    STA $62
,CF1D A0 00    LDY #$00
,CF1F B1 61    LDA ($61),Y
,CF21 C9 DA    CMP #$DA
```

```
,CF23 D0 07     BNE $cf2C
,CF25 C8        INY
,CF26 B1 61     LDA ($61),Y
,CF28 C9 80     CMP #$80
,CF2A F0 08     BEQ $cf34
,CF2C A0 02     LDY #$02
,CF2E 18        CLC
,CF2F 20 6B CE  JSR $ce6B
,CF32 90 E9     BCC $cf1D
,CF34 A0 07     LDY #$07
,CF36 38        SEC
,CF37 20 6B CE  JSR $ce6B
,CF3A A9 07     LDA #$07
,CF3C 8D 84 CF  STA $cf84
,CF3F A9 00     LDA #$00
,CF41 8D 90 CF  STA $cf90
,CF44 8D 5C CF  STA $cf5C
,CF47 A0 FF     LDY #$FF
,CF49 B1 2D     LDA ($2D),Y
,CF4B 8D 5E CF  STA $cf5E
,CF4E C8        INY
,CF4F B1 2D     LDA ($2D),Y
,CF51 85 65     STA $65
,CF53 C8        INY
,CF54 B1 2D     LDA ($2D),Y
,CF56 85 66     STA $66
,CF58 20 72 CF  JSR $cf72
,CF5B A0 FF     LDY #$FF
,CF5D C0 FF     CPY #$FF
,CF5F F0 0B     BEQ $cf6C
,CF61 91 65     STA ($65),Y
,CF63 EE 5C CF  INC $cf5C
,CF66 AA        TAX
,CF67 D0 EF     BNE $cf58
,CF69 C8        INY
,CF6A D0 F1     BNE $cf5D
,CF6C 68        PLA
,CF6D A8        TAY
,CF6E 68        PLA
,CF6F AA        TAX
,CF70 68        PLA
,CF71 60        RTS
,CF72 A0 00     LDY #$00
,CF74 AE 84 CF  LDX $cf84
,CF77 B1 61     LDA ($61),Y
,CF79 C8        INY
,CF7A 6A        ROR
,CF7B 51 61     EOR ($61),Y
,CF7D 3D DF CE  AND $ceDF,X
,CF80 51 61     EOR ($61),Y
,CF82 B8        CLV
,CF83 50 00     BVC $cf85
,CF85 2A        ROL
,CF86 2A        ROL
,CF87 2A        ROL
,CF88 2A        ROL
,CF89 2A        ROL
,CF8A 2A        ROL
,CF8B 2A        ROL
,CF8C 2A        ROL
,CF8D CA        DEX
,CF8E CA        DEX
,CF8F A0 FF     LDY #$FF
,CF91 2A        ROL
,CF92 B0 14     BCS $cfA8
,CF94 30 05     BMI $cf9B
,CF96 B9 9D CE  LDA $ce9D,Y
,CF99 90 2E     BCC $cfC9
,CF9B CA        DEX
,CF9C 2A        ROL
,CF9D 30 05     BMI $cfA4
,CF9F B9 BE CE  LDA $ceBE,Y
,CFA2 B0 25     BCS $cfC9
,CFA4 A9 00     LDA #$00
,CFA6 F0 21     BEQ $cfC9
,CFA8 6A        ROR
,CFA9 CA        DEX
,CFAA CA        DEX
,CFAB CA        DEX
,CFAC C9 F8     CMP #$F8
,CFAE 10 14     BPL $cfC4
,CFB0 69 F8     ADC #$F8
,CFB2 6A        ROR
,CFB3 C9 F4     CMP #$F4
,CFB5 10 0E     BPL $cfC5
,CFB7 69 F4     ADC #$F4
,CFB9 6A        ROR
,CFBA C9 EA     CMP #$EA
,CFBC 10 08     BPL $cfC6
,CFBE 69 EA     ADC #$EA
,CFC0 6A        ROR
,CFC1 38        SEC
,CFC2 30 03     BMI $cfC7
,CFC4 CA        DEX
,CFC5 CA        DEX
,CFC6 CA        DEX
,CFC7 69 1F     ADC #$1F
,CFC9 48        PHA
,CFCA 8A        TXA
,CFCB 10 08     BPL $cfD5
,CFCD 49 F8     EOR #$F8
,CFCF E6 61     INC $61
,CFD1 D0 02     BNE $cfD5
,CFD3 E6 62     INC $62
,CFD5 8D 84 CF  STA $cf84
,CFD8 68        PLA
,CFD9 AA        TAX
,CFDA BD 7D CE  LDA $ce7D,X
,CFDD C0 0B     CPY #$0B
,CFDF F0 12     BEQ $cfF3
,CFE1 C9 5E     CMP #$5E
,CFE3 D0 06     BNE $cfEB
,CFE5 20 72 CF  JSR $cf72
,CFE8 18        CLC
,CFE9 69 20     ADC #$20
,CFEB 8E 90 CF  STX $cf90
,CFEE C9 5F     CMP #$5F
,CFF0 F0 80     BEQ $cf72
,CFF2 60        RTS
,CFF3 38        SEC
,CFF4 E9 20     SBC #$20
,CFF6 A0 20     LDY #$20
,CFF8 8C 90 CF  STY $cf90
,CFFB 60        RTS
```

Si no dispones de un ensamblador, puedes introducir el listado hexadecimal que figura más adelante. Pero en tal caso necesitas un programa en BASIC que te facilite la introducción del código. Existen algunas diferencias entre las versiones para el **Vic 20** y el **64**. Los que poseáis un **64** deberéis empezar con la siguiente línea:

```
5 POKE 53280,1:POKE 53281,1
```

En cambio los poseedores de un **Vic 20** tendréis que empezar con:

```
5 POKE 36879,25
```

El resto del programa es común para los dos ordenadores. Aquí tienes las líneas que has de introducir antes de ejecutar el programa:

```
8 HH$="0123456789ABCDEF"
10 PRINT "[SHIFT+CLR/HOME]
   [CRSR abajo][CTRL+7]
   [CTRL+9]"TAB(6);"MONITOR
   HEXA"
15 PRINT "[4*CRSR abajo]
   [CTRL+8]1) INTRODUCE EL
   CODIGO MAQUINA"
30 PRINT"[2*CRSR abajo]2)
   GUARDA LOS BYTES"
35 PRINT "[3*CRSR abajo]
   [ctrl+7][CTRL+9][2*CRSR
   dcha.]ESCRIBE LA OPCION
   (1-2)?p"
40 GET A$:IF A$<"1" OR A$>
   "2" THEN 40
50 ON VAL(A$) GOSUB 100,300
60 GOTO 10
```

```
100 SA$="":INPUT "[SHIFT+
    CLR/HOME]DIRECCION
    INICIAL ";SA$:PRINT
    "[SHIFT+CLR/HOME]"
102 D$=LEFT$(SA$,2):GOSUB
    400:S1=C:IF W<2 THEN 100
104 D$=RIGHT$(SA$,2):GOSUB
    400:IF W<2 THEN 100
105 SA=S1*256+C
110 GOSUB 450:DD$="":INPUT
    DD$:IF DD$="" THEN 10
120 ZZ=0:D$="":DD$=DD$+" ":
    FOR LL=1 TO LEN(DD$):IF
    MID$(DD$,LL,1)=" " THEN
    130
122 D$=D$+MID$(DD$,LL,1):
    GOTO 160
130 ZZ=ZZ+1:GOSUB 400:IF W<2
    THEN PRINT:GOTO 110
150 POKE SA,C:D$="":SA=SA+1
160 NEXT LL:GOTO 110
300 CLR:HH$="0123456789ABCDE
    F":SA$="":INPUT"[SHIFT+
    CLR/HOME][CTRL+9]
    DIRECCION INICIAL[CTRL+0]
    ";SA$
302 D$=LEFT$(SA$,2):GOSUB
    400:AA=C:IF W<2 THEN 300
304 D$=RIGHT$(SA$,2):GOSUB
    400:A2=C:IF W<2 THEN 300
310 INPUT"[CTRL+9] DIRECCION
     FINAL [CTRL+0]";SA$
312 D$=LEFT$(SA$,2):GOSUB
    400:BB=C:IF W<2 THEN 310
314 D$=RIGHT$(SA$,2):GOSUB
    400:B2=C:IF W<2 THEN 310
315 INPUT "[CTRL+9] NOMBRE
    FICHERO [CTRL+0]";N$:IF
    N$="" THEN 315
316 D=1:INPUT "[CTRL+9](D)
    ISCO,(C)INTA[CTRL+0]";D$
    :IF D$="D" THEN N$="@:"+
    N$:D=8
320 PRINT"[SHIFT+CLR/HOME]P
    [SHIFT+0]44,";AA;":P
    [SHIFT+0]43,";A2
330 PRINT"[2*CRSR abajo]P
    [SHIFT+0]46,";BB;":P
    [SHIFT+0]45,";B2
335 PRINT"[2*CRSR abajo]SAVE
    ";CHR$(34);N$;CHR$(34);"
    ,";D;",1"
340 PRINT "[9*CRSR abajo]P
    [SHIFT+0]44,";PEEK(44);"
    :P[SHIFT+0]43,";PEEK(43)
350 PRINT "[2*CRSR abajo]P
    [SHIFT+0]46,";PEEK(46);"
    :P[SHIFT+0]45,";PEEK(45)
    :PRINT "[2*CRSR abajo]
    RUN[CLR/HOME]"
399 END
400 W=0:FOR Z=1 TO 16:IF
    LEFT$(D$,1)=MID$(HH$,Z,1
    )THEN W=W+1
410 IF RIGHT$(D$,1)=MID$(HH$
    ,Z,1)THEN W=W+1
420 NEXT:IFW<2 THEN RETURN
430 A=ASC(D$)-48:B=ASC
    (RIGHT$(D$,1))-48
440 C=B+7*(B>9)-(LEN(D$)=2)*
    (16*(A+7*(A>9))):RETURN
450 S1=INT(SA/256):S2=SA-S1*
    256:S(1)=INT(S1/16):S(2)
```

```
    =S1-S(1)*16
453 S(3)=INT(S2/16):S(4)=S2-
    S(3)*16
455 FOR L=1 TO 4
460 PRINT MID$(HH$,S(L)+1,1);
480 NEXT L:PRINT" ";:RETURN
```

En el capítulo anterior tienes las direcciones iniciales y las modificaciones necesarias (si no tienes un **64**), con los POKEs correspondientes a las dimensiones de la memoria del **Vic**. Los usuarios de este ordenador deben cambiar además los números del listado hexadecimal que figuran subrayados; para ver la manera de hacerlo consulta de nuevo la lista de direcciones.

Este listado corresponde al C-64. Las letras en minusculas deberán ser sustituidas en el caso del Vic-20 según la tabla presentada en el anterior capitulo.

```
:cd14 A2 1F 48 38 FD 7D ce F0
:cd1C 22 C9 80 F0 04 C9 20 D0
:cd24 0B 8A 48 A9 5E 20 14 cd
:cd2C 68 AA 10 0F C9 E0 D0 06
:cd34 8A 48 A9 5F D0 EF 68 CA
:cd3C 10 D8 60 68 8A A0 FF C9
:cd44 0A F0 03 8D 42 cd AE 99
:cd4C cd CA CA D9 9D ce D0 04
:cd54 A9 00 F0 31 CA D9 BE ce
:cd5C D0 04 A9 40 D0 27 C9 01
:cd64 10 04 A9 60 10 1F CA CA
:cd6C CA CA CA 69 DF C9 F8 10
:cd74 14 E8 0A 69 07 C9 F0 10
:cd7C 0C E8 0A 69 0F C9 C8 10
:cd84 04 E8 0A 69 37 C0 0B D0
:cd8C 05 A0 20 8C 42 cd 8E B5
:cd94 cd AE 99 cd D0 00 6A 6A
:cd9C 6A 6A 6A 6A 6A 6A 48 2A
:cdA4 3D E0 ce A0 00 11 61 91
:cdAC 61 C8 68 3D E8 ce 91 61
:cdB4 A9 FF 10 08 49 F8 E6 61
:cdBC D0 02 E6 62 8D 99 cd A9
:cdC4 00 60 48 98 48 A0 08 8C
:cdCC 1A ce A0 00 8C 13 ce B1
:cdD4 2D C8 C9 5A D0 06 B1 2D
:cdDC C9 80 F0 06 98 69 06 A8
:cdE4 D0 ED C8 8C 2F ce 68 A8
:cdEC 68 60 48 8A 48 98 48 A5
:cdF4 2F 85 61 A5 30 85 62 A0
:cdFC 00 B1 61 C9 DA D0 07 C8
:ce04 B1 61 C9 80 F0 08 A0 02
:ce0C 18 20 6B ce 90 E9 A9 FF
:ce14 A0 08 91 61 88 A9 FF 91
:ce1C 61 38 20 6B ce A9 07 8D
:ce24 99 cd A9 00 8D 42 cd 8D
:ce2C 40 ce A0 FF B1 2D 8D 42
:ce34 ce C8 B1 2D 85 65 C8 B1
:ce3C 2D 85 66 A0 FF C0 FF F0
:ce44 0A B1 65 20 14 cd EE 40
:ce4C ce D0 F0 20 14 cd A9 06
:ce54 cd 99 cd A5 61 E5 63 8D
:ce5C 1A ce A5 62 E5 64 8D 13
:ce64 ce 68 A8 68 AA 68 60 A5
:ce6C 61 85 63 71 61 85 61 C8
:ce74 A5 62 85 64 71 63 85 62
:ce7C 60 20 41 53 4F 54 52 49
:ce84 4C 45 43 5E 5F 55 4D 50
:ce8C 57 59 4E 42 47 44 46 56
:ce94 48 4B 51 58 4A 5A 5B 5C
:ce9C 5D 0A 11 04 11 17 08 11
:ceA4 08 05 17 02 07 11 08 03
:ceAC 01 01 14 08 08 08 04 08
:ceB4 08 08 0C 0E 0C 08 0B 0B
:ceBC 00 0B 04 04 08 15 06 03
:ceC4 04 06 14 03 04 11 02 01
:ceCC 08 17 03 04 07 17 06 03
:ceD4 06 01 06 1A 04 03 06 0A
:ceDC 1E 0B 0A 00 01 03 07 0F
:ceE4 1F 3F 7F FF FE FC F8 F0
:ceEC E0 C0 80 00 48 98 48 A0
:ceF4 00 B1 2D C8 C9 5A D0 06
:ceFC B1 2D C9 80 F0 06 98 69
:cf04 06 A8 D0 ED C8 8C 48 cf
:cf0C 68 A8 68 60 48 8A 48 98
:cf14 48 A5 2F 85 61 A5 30 85
:cf1C 62 A0 00 B1 61 C9 DA D0
:cf24 07 C8 B1 61 C9 80 F0 08
:cf2C A0 02 18 20 6B ce 90 E9
:cf34 A0 07 38 20 6B ce A9 07
:cf3C 8D 84 cf A9 00 8D 90 cf
:cf44 8D 5C cf A0 FF B1 2D 8D
:cf4C 5E cf C8 B1 2D 85 65 C8
:cf54 B1 2D 85 66 20 72 cf A0
:cf5C FF C0 FF F0 0B 91 65 EE
:cf64 5C cf AA D0 EF C8 D0 F1
:cf6C 68 A8 68 AA 68 60 A0 00
:cf74 AE 84 cf B1 61 C8 6A 51
:cf7C 61 3D DF ce 51 61 B8 50
:cf84 00 2A 2A 2A 2A 2A 2A 2A
:cf8C 2A CA CA A0 FF 2A B0 14
:cf94 30 05 B9 9D ce 90 2E CA
:cf9C 2A 30 05 B9 BE ce B0 25
:cfA4 A9 00 F0 21 6A CA CA CA
:cfAC C9 F8 10 14 69 F8 6A C9
:cfB4 F4 10 0E 69 F4 6A C9 EA
:cfBC 10 08 69 EA 6A 38 30 03
:cfC4 CA CA CA 69 1F 48 8A 10
:cfCC 08 49 F8 E6 61 D0 02 E6
:cfD4 62 8D 84 cf 68 AA BD 7D
:cfDC ce C0 0B F0 12 C9 5E D0
:cfE4 06 20 72 cf 18 69 20 8E
:cfEC 90 cf C9 5F F0 80 60 38
:cfF4 E9 20 A0 20 8C 90 cf 60
```

UTILIZA TU COMPRESOR DE TEXTOS

- AHORRO DE MEMORIA
- CODIFICACION DE LAS SENTENCIAS PRINT
- FUNCIONAMIENTO DE LA RUTINA DE DECODIFICACION

Sácale punta al lápiz y ponte a trabajar en tu obra maestra. Con un compresor de textos que funcione a tope ya puedes escribir la aventura épica con la que siempre soñaste.

En este momento dispones de un programa en código máquina almacenado en cinta. Tanto si has arrancado a partir del listado en lenguaje ensamblador, como si lo has hecho partiendo de hexadecimal, el programa es exactamente el mismo. Pero para poder utilizar el código máquina tienes que escribir algo en BASIC.

En esta parte del coleccionable, veremos la manera de utilizar el BASIC con el compresor de textos en código máquina y cómo incorporarlo en un juego de aventuras.

DESARROLLO DE JUEGOS DE AVENTURAS

Hemos visto anteriormente la manera de escribir juegos de aventuras. Con unas pequeñas modificaciones de los programas, puedes utilizar el compresor de textos para ahorrar una gran cantidad de memoria.

Para empezar, escribe tu aventura con arreglo a las directrices dadas anteriormente, pero procura incluir mensajes que sean cortos, ya que más adelante los eliminaremos. Por ejemplo, a lo mejor quieres escribir algo como ACABAS DE LLEGAR A UN CLARO DEL BOSQUE. HAY UNA PATA DE PALO PARCIALMENTE OCULTA POR UN MANTEL. En vez de teclear todo esto, utiliza en

la sentencia PRINT algo más breve, por ejemplo CLARO. Haz lo mismo en todas las sentencias PRINT, para que puedas comprobar que tu aventura realmente funciona, depurándola y cerciorándote de que no tiene errores lógicos graves.

Cuando estés satisfecho con el propio juego, puedes pasar a utilizar el compresor de textos y escribir de nuevo las sentencias PRINT.

Lo primero que debes hacer es decidir lo que realmente tienen que decir tus mensajes. Si dispones de una impresora, el trabajo puede resultar más sencillo porque puedes listar el programa y trabajar sobre el listado. Escribe una lista de mensajes junto con los números de línea correspondientes a cada mensaje. Toda sentencia PRINT que contenga más de dos o tres palabras, te va a merecer la pena codificarla. En los mensajes más cortos es poco o nada el espacio de memoria que se ahorra con la codificación, por lo que puedes ahorrar el trabajo de hacerla.

FUNCIONAMIENTO

Al codificar el texto, el programa en código máquina almacena la información en binario en una matriz llamada Z%. Hay otra matriz, el A%, que contiene los punteros que permiten a la máquina saber en qué parte de Z% comienza cada uno de los mensajes.

Cuando quieras decodificar, es decir, escribir en pantalla alguno de los mensajes, se utilizan los punteros que hay almacenados en A% para extraer la sección correcta de información en binario con la que alimentar la rutina de decodificación. A medida que va siendo decodificada la información en binario, los caracteres se van cargando en una cadena de caracteres llamada Z$, que previamente ha sido definida con una longitud igual a la del mensaje más largo. Para que aparezca el mensaje en pantalla, no tienes más que pedir al ordenador que ejecute PRINT Z$ y suministrarle el número adecuado de A% que le diga la parte del mensaje que tiene que cargar en Z$.

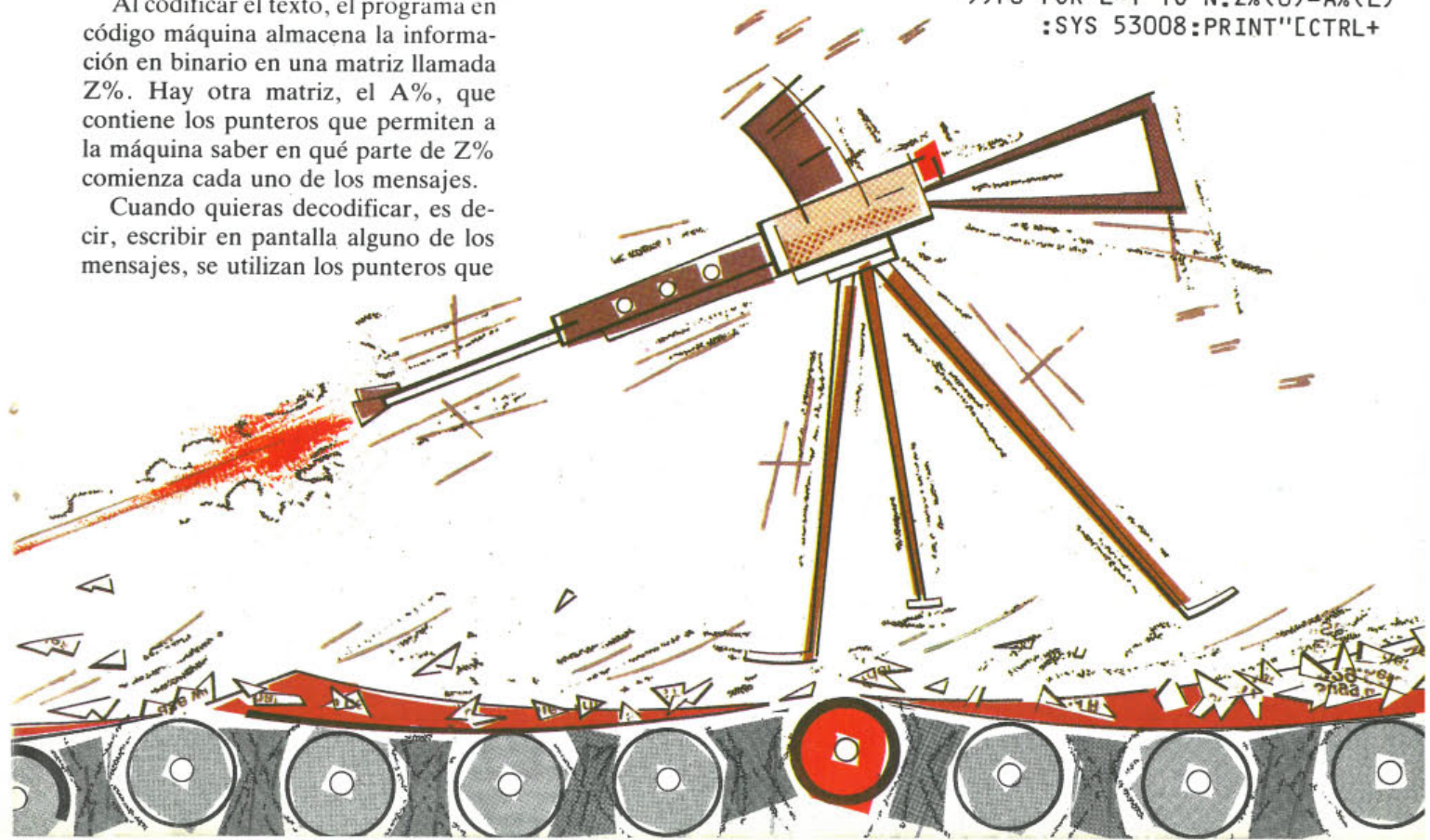
Cuando tengas la aventura con los mensajes cortos funcionando a tu entera satisfacción, puedes introducir las siguientes líneas. Las llamadas a SYS corresponden al **Commodore 64**, pero aquí tienes los números que hay que cambiar en el caso del **Vic 20** para las distintas expansiones:

C-64	52718	52678	52976	53008
8 K	15854	15814	16112	16144
16 K	24046	24006	24304	24336
24 K	32238	32198	32496	32528
32 K	36334	36294	36592	36624

Además en el **Vic 20** tienes que modificar la línea 9905, cambiándola por la siguiente:

Teclea

```
9900 INPUT"[SHIFT+CLR/HOME]
     CUANTOS MENSAJES";N:IF
     N=0 THEN 9900
9905 DIM A%(N):T=INT(FRE(0)/
     2.01):IF T-80<0 THEN
     PRINT"NO HAY ESPACIO":
     END
9910 DIM Z%(T):PRINT"[SHIFT+
     CLR/HOME]":Z$="PRUEBA":
     SYS 52678:FOR L=1 TO N
9930 Z$="":PRINT"[CRSR abajo]
     [ctrl+9]ESCRIBE EL
     MENSAJE";L:INPUT Z$:IF
     Z$=""THEN N=L-1:GOTO
     9960
9940 PRINT:SYS 52718:A%(L)=
     Z%(0):IF Z%(T-80)<>0
     THEN N=L:GOTO 9960
9950 NEXT L
9960 Z$="1234567890123456789
     0123456789012345678901":
     Z$=Z$+Z$
9965 SYS 52976:PRINT"[SHIFT+
     CLR/HOME]"
9970 FOR L=1 TO N:Z%(0)=A%(L)
     :SYS 53008:PRINT"[CTRL+
```



```
     9]MENSAJE";L:PRINT Z$
9980 NEXT L:PRINT "[CRSR
     abajo][ctrl+9]LO GUAR
     DAS EN CINTA (S/N)?"
9990 GET A$:IF A$<>"S" AND
     A$<>"N" THEN 9990
9995 IF A$="N" THEN RUN
10000 INPUT "ESCRIBE EL
     NOMBRE ";N$:OPEN1,1,1,
     N$:PRINT#1,N
10010 FOR L=1 TO N:PRINT#1,
     A%(L) CHR$(44):NEXT L
10015 FOR L=1 TO T:IF Z%(L)
     <>0 THEN NEXT L
10020 T=L:PRINT#1,T:FOR L=1
     TO T:PRINT#1,Z%(L)CHR$
     (44):IF Z%(L)<>0 THEN
     NEXT L
10030 CLOSE 1
```

Ejecuta el programa y responde a los mensajes que aparezcan en pantalla; si alguno de estos mensajes tiene que ir formateado para ocupar más de una línea, no utilices la tecla de retorno de carro RETURN , ya que el programa lo interpretaría como un final de mensaje. En vez de hacer eso, completa las líneas con espacios; no te preocupes, los espacios no consumen mucha memoria.

Cuando hayas terminado de introducir el texto, aparecerá un mensaje diciéndote que lo almacenes en cinta. Borra todas las líneas a partir de la 9900 y añade estas líneas al principio del programa:

```
10 CLR:INPUT "ESCRIBE EL
   NOMBRE ";N$:OPEN 1,1,0,N$
   :INPUT#1,N:DIM A%(N)
20 FOR L=1TO N:INPUT#1,A%(L)
   :NEXT L
30 INPUT#1,T:DIM Z%(T):FOR
   L=1 TO T:INPUT#1,Z%(L):IF
   Z%(L)<>0 THEN NEXT L
40 CLOSE 1
50 Z$="12234567890123456789
   0123456789012345678901":Z$=
   Z$+Z$
60 SYS 52976
```

Teclea además las siguientes líneas

```
9899 END
9900 Z%(0)=A%(L):SYS53008:
     PRINT Z$:RETURN
```

No te olvides de cambiar los SYS en el caso del **Vic**.

A continuación, cada una de las sentencias PRINT para las que haya un texto comprimido ha de ser sustituída por un GOSUB 9900. Por ejemplo, si la línea 500 contiene el quinto mensaje, cambia:

```
500 PRINT "MENSAJE QUINTO"
```

por

```
500 L=5:GOSUB 9900
```

L es una variable a la que se asigna un valor igual al número del mensaje. En la subrutina se extrae el elemento L-simo de la matriz A%, que son los punteros válidos para seleccionar la parte correcta de Z%. Cuando quieras jugar el juego, carga primero el compresor de textos en la memoria. Teclea:

```
LOAD "nombre del compresor
de texto",1,1
```

A continuación haz un NEW. Después carga tu aventura con LOAD y ejecútala. Tienes que ocuparte de cargar el texto comprimido y asegurarte de que tienes lista la cinta que contiene dicho texto comprimido.

PREDICIENDO LO IMPREDECIBLE

- LAS REGLAS DEL JUEGO
- SACAR LOS NUMEROS DE UNA CAJA O UN SOMBRERO
- NUMEROS ALEATORIOS
- MUESTRAS Y MUESTREO

¿Qué posibilidades tienes de acertar un pleno en las quinielas de esta semana? Puede que no mejores tu suerte, pero al menos entenderás porqué pierdes.

La instrucción de un novato en el manejo de un moderno avión comercial o de un bombardero sería enormemente cara al tener en cuenta factores tales como el combustible, las pistas de aterrizaje y toda la infraestructura de apoyo. Por eso las autoridades militares opinan que es más barato gastar dinero en aviones de entrenamiento y simuladores, enormes artefactos controlados por ordenador que nunca abandonan el suelo, pero que dan una impresión de vuelo de lo más realista.

Este mismo principio resulta cierto en un gran número de circunstancias, incluída la programación de juegos, el desarrollo industrial y el *marketing*, así como la investigación científica y los estudios gubernamentales. Es preferible programar un ordenador para predecir, por ejemplo, el resultado probable de un fallo estructural o una determinada política económica, que realizar el experimento real que podría requerir cinco años o más. No hay que sorprenderse pues de que la simulación sea una actividad que resulta especialmente impulsada. Pero debido a su complejidad sólo ha podido ser posible con la llegada de los ordenadores. Actualmente, hasta los micros domésticos pueden realizar un cierto grado de simulación.

Una de las principales herramientas de este tipo de creación de modelos por ordenador es el conjunto de reglas que nos proporciona el estudio matemático de la probabilidad.

El resto del modelo se deriva de un análisis de la información estadística, obtenido a partir de una descripción de la situación real.

Con unas cuantas reglas sencillas, el usuario del microordenador doméstico puede predecir las alternativas futuras para un sinfín de sucesos diferentes, si bien la fiabilidad de los resultados depende de la precisión de los datos introducidos en el programa o de la precisión de las reglas por las que

se rige el juego. Se puede demostrar este hecho mediante la simulación de los resultados de los partidos de fútbol.

Todos los fines de semana de la temporada futbolística, millones de aficionados a las quinielas están pendientes de los resultados a la espera de conseguir algún premio sustancioso. Por desgracia, todos menos unos pocos se ven decepcionados y tienen que esperar una semana más para tener otra oportunidad. Si no quieres tener que esperar una semana entera para tener otra oportunidad, teclea el siguiente programa y prueba tu suerte todas las veces que quieras:

Teclea para C-64 y Vic-20

```
20 DIM A$(2,14),R$(4)
50 PRINT "[SHIFT+CLR/HOME]
   [CRSR abajo]>[CRSR dcha.]
   [CTRL+9]PREVISION
   FUTBOLISTICA"
100 PRINT "[SHIFT+CLR/HOME]
    [CRSR abajo]>[CTRL+9]
    [CRSR dcha.]ESCRIBE LOS
    ENCUENTROS"
110 FOR D=1 TO 2000:NEXT D
120 FOR K=1 TO 14
130 PRINT"[SHIFT+CLR/HOME]"
140 PRINT "EQUIPO LOCAL";K;
    " ";:INPUT A$(1,K)
150 PRINT "EQUIPO VISITANTE"
    ;K;" ";:INPUT A$(2,K)
160 NEXT K
170 PRINT "[SHIFT+CLR/HOME]
    [CRSR abajo]>[CTRL+9]
    [CRSR dcha.]PARTIDOS
    DISPUTANDOSE"
180 FOR V=1 TO 2000:NEXT V
190 PRINT "[SHIFT+CLR/HOME]
    [CRSR abajo]>[CTRL+9]
    [CRSR dcha.]RESULTADOS"
    TAB(25)"GANA"
200 R$(1)="EQUIPO LOCAL":
    R$(2)="EQUIPO VISITANTE"
210 R$(3)="EMPATE":
220 PRINT"[CRSR abajo]
    PARTIDO":FOR I=1 TO 14
230 Y=RND(1)
240 IF Y<=0.5 THEN Z$=R$(1)
250 IF Y>0.5 AND Y<=0.7 THEN
    Z$=R$(2)
260 IF Y>0.7 THEN Z$=R$(3)
280 PRINT A$(1,I);"-";A$
    (2,I)TAB(20)Z$
290 NEXT I
310 PRINT "[CRSR abajo]OTRA
    PARTIDA (S/N)?";
320 GET T$:IF T$="N" THEN
    PRINT"":END
330 IF T$="S" THEN RUN
340 GOTO 320
```

Aunque en esta simulación futbolística no hay reparto de dividendos, puede que te comunique una sensación bastante realista. En cualquier caso servirá para demostrar lo pequeña que es la posibilidad de conseguir 24 puntos.

Ejecuta el programa.

El mal tiempo obliga a veces a suspender determinados partidos. Cuando esto sucede, un grupo de expertos futbolísticos decide qué resultados se habrían producido de haberse jugado realmente el encuentro. De hecho, tanto el programa como el grupo de expertos son simuladores; proporcionan una representación simbólica de un proceso real.

El programa no es especialmente difícil. Una vez introducidos los encuentros, la parte principal de la simulación decide el resultado de cada partido y lo presenta en pantalla.

DECIDIENDO LOS RESULTADOS

La regla en que se basa el ordenador para decidir cualquier resultado, es la vieja regla de las quinielas de que la mitad de los resultados son victorias domésticas, las victorias fuera de casa y los empates son el resto. Esta proporción puede variar en las líneas 240, 250 y 260.

GENERACION DE NUMEROS ALEATORIOS

Hace años se utilizaban métodos manuales o mecánicos, tales como lanzar dados, barajar cartas o emplear una ruleta, para generar listas de números aleatorios. Estos procedimientos eran lentos y tediosos, por lo que el famoso matemático norteamericano **John Von Neuman** propuso la técnica del cuadrado medio. Se empieza con un número de cuatro cifras (la semilla), el siguiente número aleatorio se obtiene multiplicando la semilla por sí misma y quedándose con los cuatro dígitos centrales. Por ejemplo, supongamos que la semilla es 5272. El segundo número se genera tomando los cuatro dígitos centrales de (5272 ↑ 2) que es 27 793 984. El resultado (7939) es prácticamente aleatorio. Se puede obtener un segundo número elevando al cuadrado el número 7939, continuando así el proceso.

Naturalmente, surge la pregunta:

¿Cómo puede un proceso matemático, que es esencialmente repetitivo, producir números verdaderamente aleatorios? La respuesta es que en realidad no puede hacerlo. Sin embargo los números así obtenidos se comportan como si realmente fueran aleatorios, por lo que generalmente se les llama pseudoaleatorios o quasi-aleatorios. De hecho los números generados utilizando la función RND son pseudoaleatorios. Lástima que esta técnica del cuadrado central no sea muy útil para la generación de números aleatorios por ordenador.

Aparte de ser lenta, la secuencia se repite en seguida y en cuanto se obtiene un cero se repite todo el proceso. La mayoría de los micros domésticos utilizan un método de **congruencias**, que utiliza los restos para generar secuencias pseudoaleatorias. El siguiente programa utiliza una sencilla fórmula y la función INT (parte entera) para proporcionar un ejemplo de esta técnica:

Teclea para C-64 y Vic-20

```
30 PRINT"[SHIFT+CLR/HOME]
   [CRSR abajo]>[CTRL+9]
   [CRSR dcha.]NUMEROS
   SEUDOALEATORIOS[CRSR
   abajo]"
40 PRINT"CUANTOS
   NUMEROS": INPUT N
50 X=.677829*TI/60
60 X=X*1842.95
70 X=X-INT(X)
80 P=INT(X*1000):PRINT
   .001*P,
90 S=S+1
110 IF S<=N THEN 60
```

La línea 50 utiliza la función de hora del ordenador para establecer como punto de partida un número diferente en cada ejecución. El número .677829 es una constante arbitraria. Después de multiplicar el valor de partida por otra constante (línea 60), se obtiene el resto de la parte decimal. Cambia el valor de las constantes de las líneas 50 y 60 y ejecuta nuevamente el programa para ver qué tipo de resultados obtienes.

En muchos casos es preferible —y bastante más sencillo— utilizar la función RND que lleva incorporada tu ordenador. Al modificar el valor de x en la expresión RND(x) podrás seleccionar una secuencia que se repita, lo cual es una buena prueba para reinicializar la «semilla» cada vez.

Quizá el punto más importante que tienes que recordar cuando escribas programas es que la función RND lo que te da es una variable aleatoria y no un valor algebraico

MUESTRAS Y OBSERVACIONES

Los sondeos preelectorales, las organizaciones de consumidores y los gobiernos utilizan los ordenadores para generar muestras aleatorias. Por ejemplo, es importante que una empresa dedicada a la investigación de mercados y entrevista a 1000 personas, obtenga una muestra típicamente variada, que pueda representar a un grupo mucho mayor de población. No sería válido por ejemplo elegir a todos los individuos entre los miembros de un club de coches antiguos, cuando lo que se pretende investigar son las costumbres nacionales en materia de preferencias automovilísticas.

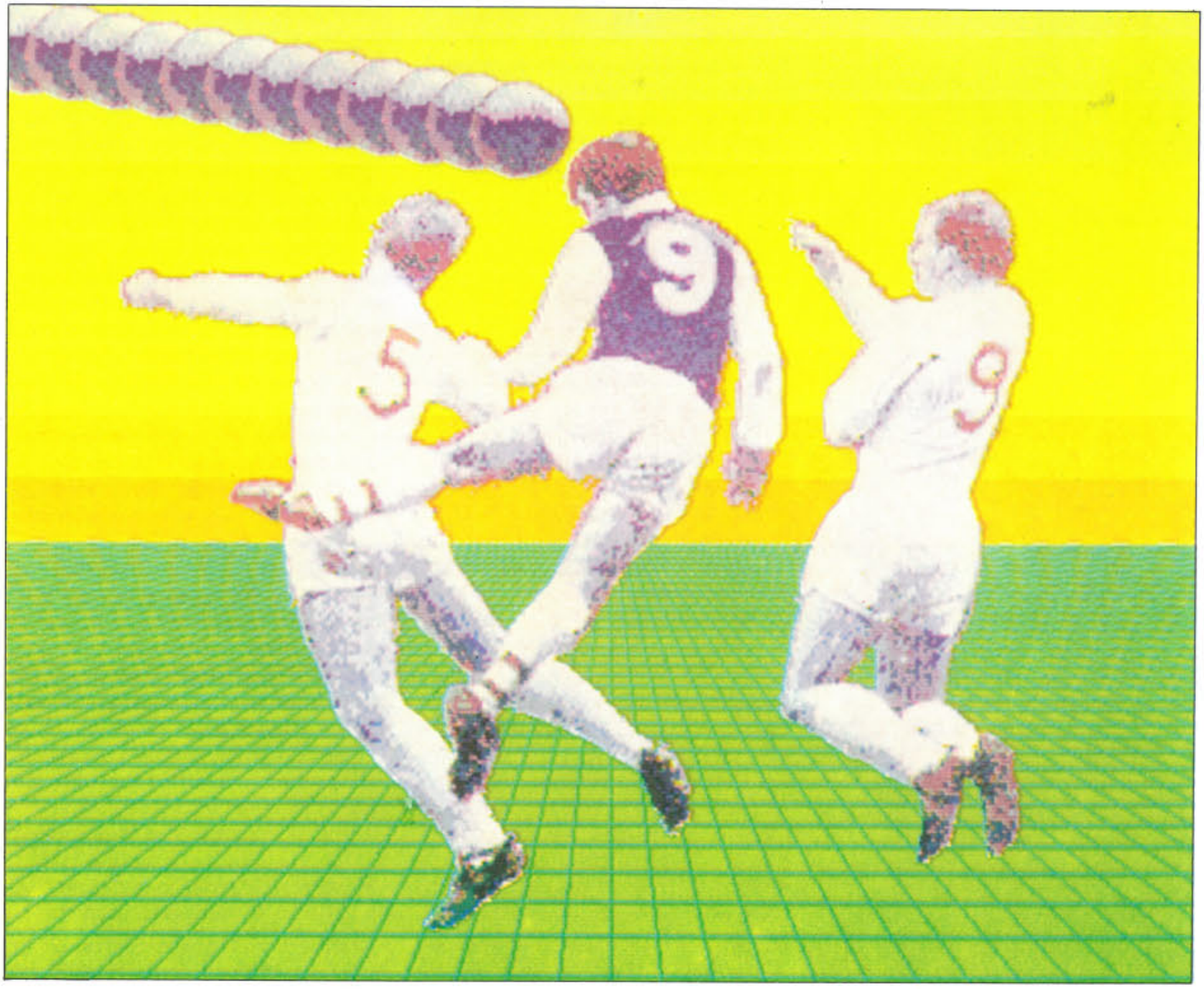
La mejor forma de hacer que una muestra sea representativa es obtenerla aleatoriamente, con lo que se excluye todo tipo de sesgo o tendencia.

Esto sigue siendo cierto también cuando se trabaja con una muestra más pequeña, como podría ser el caso de un club de amigos de los coches antiguos o de los lectores de **INPUT**. También en este caso es importante la selección aleatoria, por ejemplo en el caso en que quisieras determinar las preferencias de los lectores de las revistas de informática.

También puede ser que quieras extraer una muestra de tu propia lista de miembros de un club informático. El proceso de muestreo es básicamente similar a la simulación de una extracción de papeletas de un sombrero. No obstante en la simulación, cada papeleta vuelve a ser introducida en el sombrero antes de proceder a la siguiente extracción. En el muestreo, cada papeleta seleccionada se deja fuera del sombrero y se realiza la siguiente extracción. Teclea el siguiente programa para ver cómo se utiliza RND para generar una muestra aleatoria:

Teclea para C-64 y Vic-20

```
20 DIM A$(10)
50 PRINT "[SHIFT+CLR/HOME]
   [CRSR abajo]>[CTRL+9]
   [CRSR dcha.]MUESTREO
   ALEATORIO[CRSR abajo]"
100 FOR I=1 TO 10:READ A$(I)
    :NEXT I
110 INPUT "TAMANO DE LA
   MUESTRA";N:PRINT
120 FOR V=1 TO 10:B$(V)=A$
```



```
    (V):NEXT V
130 FOR J=1 TO N
140 R=1+INT(RND(1)*10)
150 IFB$(R)="" THEN 140
160 PRINT B$(R)
170 B$(R)=""
180 NEXT J
190 PRINT "[CRSR abajo]
    [CTRL+7]OTRA MUESTRA
    (S/N)?"
200 GET G$:IF G$="S" THEN
    RUN
210 IF G$<>"N" THEN 200
215 PRINT "[SHIFT+CLR/HOME]"
220 DATA BONN,COPENAGUE,
    LONDRES
230 DATA MADRID,MOSCU,NUEVA
    YORK
240 DATA PARIS,ROMA,
    ESTOCOLMO,VIENA
```

Después de leer los datos (línea 100) se genera un número entero aleatorio R comprendido entre 1 y 10 (línea 140). A continuación tu ordenador imprimirá el elemento R-ésimo de la lista (línea 140). Una vez que ha sido seleccionado un artículo, hay que eliminarlo del banco de datos para que no resulte elegido por segunda vez. La línea 170 se ocupa de esto. Observa que podrías utilizar RND(10) para generar una variable aleatoria, si bien el método de la línea 140 se deriva del uso de las fracciones decimales para probar el margen de probabilidades.

Normalmente, el colectivo de datos del que se obtiene la muestra es bastante más extenso que los diez elementos de los que consta este ejemplo. Para muestrear semejantes volúmenes de datos, el programa anterior sería lento e ineficiente. Supongamos por ejemplo que un encuestador desea extraer 200 nombres de un censo electoral de 60000 votantes de una determinada circunscripción electoral; un programa como el anterior necesitaría buscar en toda la lista de electores 200 veces. Para evitar la larga espera que esto llevaría aparejado, sería mucho mejor utilizar un método que permitiera realizar la selección en un solo paso.

LA BUSQUEDA DE UN SOLO PASO

En el método de un solo paso se leería toda la lista electoral de arriba a abajo una sola vez. Al ir considerando cada nombre, se toma la decisión de si incluir o no a esa persona en la muestra. Este método resulta fácil de programar; realiza los siguientes cambios en el último programa y vuelve a ejecutarlo:

Teclea para C-64 y Vic-20

```
50 PRINT "[SHIFT+CLR/HOME]
   [CRSR abajo]>[CTRL+9]
   [CRSR dcha.]PASADA
   SENCILLA"
60 PRINT"[2*CRSR dcha.]
   [CTRL+9]MUESTREOS
   ALEATORIOS[2*CRSR abajo]"
120 A=N:C=10
130 FOR J=1 TO 10
140 IF A=0 THEN 190
150 IF RND(1)<=A/C THEN
    PRINT A$(J):GOTO 170
160 C=C-1:GOTO 180
170 A=A-1:C=C-1
```

Si ahora tecleas un 3, para seleccionar tres elementos de la lista, el primero de ellos (Bonn) es el primero que se considera. Si la función RND (línea 150) es menor que 3/10, será seleccionado Bonn. Copenague es el seleccionado a continuación. Si Bonn ya está en la selección, Copenague sólo podrá ser seleccionado si la función RND genera un valor menor que 2/9. Por otra parte, si todavía no ha sido seleccionado Bonn, las probabilidades de Copenague se elevarán hasta 3/9. Las líneas 160 y 170 actualizan las probabilidades. Cuando compares los resultados de las selecciones de este programa con las del anterior, advertirás que el método de un solo paso da las muestras en orden alfabético.

A primera vista podría parecer que con una lista de sólo diez ciudades, el número de posibles muestras es pequeño. Esto no es cierto. De hecho hay 120 posibles muestras diferentes de tamaño 3 y 252 muestras de tamaño 5. En un próximo artículo veremos de qué forma pueden desarrollarse estas ideas para ser utilizadas en el tipo de simulación conocida como modelización.

HACIENDO PREDICCIONES

Las técnicas de simulación que se describen en este artículo contienen muchos de los elementos que necesitarías para disponer de un método que te permitiera predecir los resultados de un juego. Por ello te resultará fácil modificar los programas o utilizar determinadas secciones de los mismos configuradas a la medida de tus necesidades. Por ejemplo, podrías escribir un programa que decidiera qué partidos tendrán como resultado un empate, seleccionar a continuación algunos de estos encuentros de forma aleatoria y comparar tus predicciones con los resultados reales. Esto te requeriría el uso de secciones de los programas anteriores, unidas mediante unas líneas que hagan de enlace. Un posible uso de este programa sería evitar la tendencia de una persona a elegir los partidos del boleto con arreglo a la separación de los números; suelen distribuir sus preferencias de forma casi regular en el boleto. De hecho, sería interesante comparar ambos métodos para elegir cuál es el que da los resultados más apropiados.

EL ZOCO DE INPUT

Todo se compra y se vende. Los antiguos zocos fueron lugares destinados a todo tipo de transacciones. INPUT también tiene el suyo. Vuestras operaciones de compra, cambio o venta serán publicadas en esta sección, pero dos son las limitaciones que imponemos:

a) La propuesta tendrá que ver con la microinformática.
b) Nos reservamos el derecho de no publicar aquellos insertos de los que se sospeche un trasfondo lucrativo.

Ahora un ruego. Tratar de resumir al máximo el texto; escribir casi como un telegrama siendo claros y concisos.

Envía tu mensaje a:

INPUT COMMODORE-ZOCO
c./ Alberto Alcocer, 46
28016 **MADRID**

APROVECHA LAS INTERRUPCIONES

- EL ORDENADOR COMO RELOJ
- LA PRESENTACION DEL RELOJ
- UNA SENCILLA RUTINA EN CODIGO MAQUINA
- PUESTA EN HORA

Si dispones de un poco de tiempo, ensaya la siguiente rutina en código máquina, que te presentará el propio «reloj» interno de tu ordenador como si se tratase de un reloj digital siempre en hora

Tu ordenador dispone de un reloj interno que avanza con ritmo constante y sirve para regular sus operaciones. También puedes servirte de este reloj en un buen número de instrucciones del BASIC, tal como ocurre con PAUSE en el Simon's BASIC.

Dichas instrucciones del BASIC hacen que el ordenador se ponga a contar hasta un número especificado, en cincuentaavos de segundo. Hay otras muchas operaciones que utilizan el reloj de una forma análoga; así ocurre por ejemplo en los programas para componer y ejecutar música, en los que hay que especificar la duración de cada nota.

TOMANDO TIEMPOS

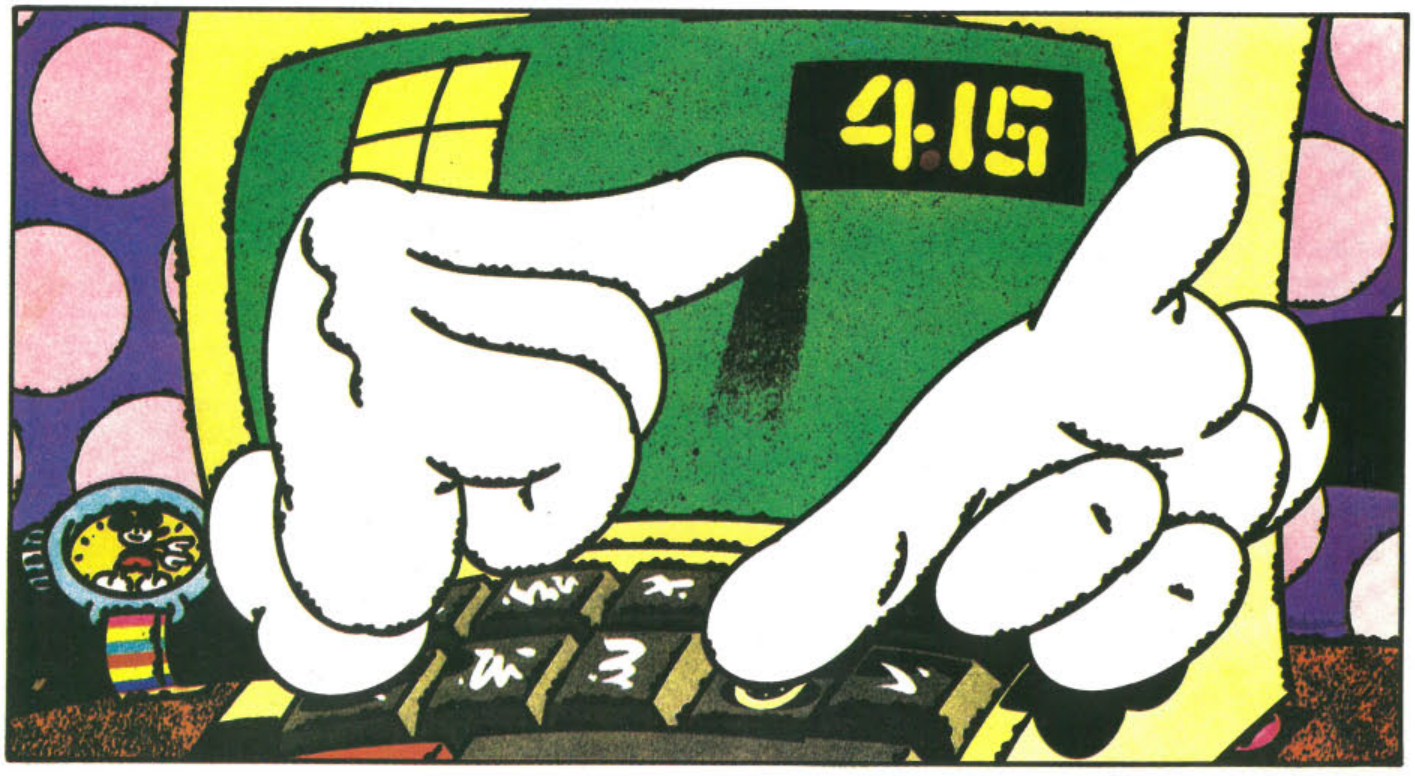
De hecho, independientemente de que tu programa especifique o no, de esta forma tan evidente, la longitud de las operaciones, el ordenador consulta su reloj interno, del que se sirve constantemente en la ejecución de los programas.

Para tí es muy sencillo que el ordenador mida tiempos. Para ello no tienes más que escribir un sencillo programa de BASIC que escriba la hora, se pare durante un segundo, sume un segundo y vuelva a imprimir de nuevo. Si pruebas a hacerlo así, en seguida descubrirás que la pausa necesaria es realmente bastante más pequeña que un segundo, debido a el tiempo que el ordenador necesita para realizar las operaciones de suma y PRINT.

Un reloj de este tipo tiene dos graves inconvenientes. El primero de ellos es que solamente mantiene la hora mientras el ordenador está conectado. Esto puede que no sea un problema muy severo; si solamente quieres saber cuánto tiempo has estado trabajando en una determinada tarea, puede incluso ser una ventaja. Pero el segundo inconveniente es bastante más serio. Consiste en que en cuanto quieras utilizar tu ordenador para cualquier otra cosa, el reloj se detiene. Naturalmente, ello se debe a que no puedes ejecutar dos programas BASIC a la vez. La solución está en la utilización de una rutina en código máquina.

UN RELOJ DE INTERRUPCIONES

Al igual que otros programas en código máquina, que han de ser ejecutados incluso cuando está corriendo un programa en BASIC, el reloj de



código máquina que veremos a continuación hace uso de una rutina activada por interrupciones.

Ya hemos visto un ejemplo de aprovechamiento de las interrupciones para ejecutar un programa trazador. De hecho el método para hacer esto es análogo en todos los ordenadores.

Durante todo el tiempo que la máquina está en operación se ve constantemente interrumpida a intervalos regulares durante una pequeña fracción de segundo. Esto ocurre incluso cuando se está ejecutando un programa en BASIC, ya que el ordenador tiene que examinar si se ha pulsado alguna tecla. Por eso el programa BASIC se detiene mientras el ordenador explora el teclado y vuelve a ejecutarse nuevamente hasta que llega el momento de la siguiente interrupción.

Puedes incluir una rutina en código máquina en esta exploración del teclado, de manera que se ejecute en las imperceptibles paradas del programa en BASIC. El resultado es que hay dos programas que parecen estar ejecutándose al mismo tiempo.

Como la propia interrupción está controlada por el reloj del ordenador, resulta ideal para nuestro objetivo, ya que se puede hacer funcionar el reloj sin más que contar el número de interrupciones. La frecuencia de las interrupciones varía de unos ordenadores a otros, pero el principio utilizado es siempre el mismo.

El programa que viene a continuación configura un sencillo reloj digital que cuenta las horas, minutos y segundos a partir del momento en que el reloj es cargado y puesto en marcha. Puedes poner a cero la lectura de la pantalla, de forma que lo puedes utilizar como reloj de tiempo real o como cronógrafo.

El tiempo dado por este reloj no es absolutamente exacto. La rutina de inicialización de cada uno de los bucles requiere un cierto tiempo, pero se trata de millonésimas de segundo. Por eso cualquier error que se presente se deberá más bien a algún desvío del reloj propio de tu ordenador. Incluso así tu reloj tendrá una deriva inferior a un segundo por día. No obstante, si utilizas las instrucciones SAVE, LOAD, BEEP, SOUND o PLAY, el reloj se parará durante el tiempo que dure la operación correspondiente. La lectura digital se presenta constantemente en la esquina superior derecha de la pantalla, superponiéndose sobre cualquier otra cosa que hubiera allí. Si esto te causa problemas, puedes reconfigurar tu pantalla de modo que se elimine la primera línea.

Teclea para Commodore-64

```
10 S=0:FOR Z=49152 TO 49267:
   READ X:S=S+X:POKE Z,X:
   NEXT Z
20 IF S<>12556 THEN PRINT
   "ERROR EN LAS DATAS":END
30 TI$="000000":FOR Z=0 TO 5
   :POKE 837-Z,VAL(MID$(TT$,
   Z+1,A)):NEXT Z
40 SYS 49152:PRINT"[SHIFT+
   CLR/HOME]CORRECTO"
100 DATA 120,169,17,141,20,3
    ,169,192,141,21,3,88,169
    ,0,133,251,96
110 DATA 230,251,165,251,201
    ,60,208,45,169,0,133,251
    ,24,162,0
120 DATA 189,64,3,105,1,157,
    64,3,201,10,208,26,169,0
    ,157,64,3
130 DATA 254,65,3,189,65,3,
    201,6,208,11,169,0,157,
    65,3,232,232
140 DATA 224,6,208,218,173,
    69,3,201,1,208,15,173,68
    ,3,201,3
150 DATA 208,8,169,0,141,68,
    3,141,69,3,160,6,162,0,
    185,63,3,105,176
160 DATA 157,32,4,169,1,157,
    32,216,232,136,208,239,
    76,49,234
```

Listado ensamblado y hexadecimal del reloj por interrupciones.

```
,C000 78       SEI
,C001 A9 11    LDA #$11
,C003 8D 14 03 STA $0314
,C006 A9 C0    LDA #$C0
,C008 8D 15 03 STA $0315
,C00B 58       CLI
,C00C A9 00    LDA #$00
,C00E 85 FB    STA $FB
,C010 60       RTS
,C011 E6 FB    INC $FB
,C013 A5 FB    LDA $FB
,C015 C9 3C    CMP #$3C
,C017 D0 2D    BNE $C046
,C019 A9 00    LDA #$00
,C01B 85 FB    STA $FB
,C01D 18       CLC
,C01E A2 00    LDX #$00
,C020 BD 40 03 LDA $0340,X
,C023 69 01    ADC #$01
,C025 9D 40 03 STA $0340,X
,C028 C9 0A    CMP #$0A
,C02A D0 1A    BNE $C046
,C02C A9 00    LDA #$00
,C02E 9D 40 03 STA $0340,X
,C031 FE 41 03 INC $0341,X
,C034 BD 41 03 LDA $0341,X
,C037 C9 06    CMP #$06
,C039 D0 0B    BNE $C046
,C03B A9 00    LDA #$00
,C03D 9D 41 03 STA $0341,X
,C040 E8       INX
,C041 E8       INX
,C042 E0 06    CPX #$06
,C044 D0 DA    BNE $C020
,C046 AD 45 03 LDA $0345
,C049 C9 01    CMP #$01
,C04B D0 0F    BNE $C05C
,C04D AD 44 03 LDA $0344
,C050 C9 03    CMP #$03
,C052 D0 08    BNE $C05C
,C054 A9 00    LDA #$00
,C056 8D 44 03 STA $0344
,C059 8D 45 03 STA $0345
,C05C A0 06    LDY #$06
,C05E A2 00    LDX #$00
,C060 B9 3F 03 LDA $033F,Y
,C063 69 B0    ADC #$B0
,C065 9D 20 04 STA $0420,X
,C068 A9 01    LDA #$01
,C06A 9D 20 D8 STA $D820,X
,C06D E8       INX
,C06E 88       DEY
,C06F D0 EF    BNE $C060
,C071 4C 31 EA JMP $EA31
```

El código máquina contiene una serie de sentencias DATA que la línea 10 se encarga de POKEar en memoria. Como hay una gran cantidad de datos y es fácil cometer errores de transcripción, la línea 10 también prevé una comprobación del total. Si la

GANADORES DE LOS MEJORES DE INPUT COMMODORE

En el sorteo correspondiente al número 11 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Juan Luis Arroyo Cedrón	Madrid	«V»
Agustí García Vivancos	Reus (Tarragona)	Green Beret
Daniel Rodríguez Pereyra	Granada	Commando
Rodrigo Espejo Ponce	Milanova del Cami (Barcelona)	Skyfox
Antonio Sarti González	Parla (Madrid)	Saboteur
Matías Fernando Gutiez	Madrid	Saboteur
Alfonso Delgado Recio	Almadén (Ciudad Real)	Fighting warrior
Esperanza Corral Rodríguez	Córdoba	Kung fu master
Galo Alfonso Sánchez	Madrid	Spindizzy
Jorge Muñoz Baños	H. de Llobregat (Barcelona)	Green Beret

suma resultante no es correcta, la línea 20 responde con un mensaje de error, invitándote a comprobar los datos.

El código máquina es llamado automáticamente en la línea 40 al ejecutar este programa en BASIC. El reloj arranca en 00:00:00 pero puedes cambiar este valor modificando el valor de TI$ (puesto a 000000 en la línea 30). El reloj se detendrá si pulsas RUN/STOP y RESTORE. Para empezar de nuevo no tienes más que teclear:

SYS 49152

El programa para el **Vic 20** es semejante al del **Commodore 64**. Aquí tienes las líneas que difieren:

Teclea para Vic-20

```
5 POKE 51,255:POKE 52,27:
  POKE 55,255:POKE 56,27:CLR
10 S=0:FOR Z=7168 TO 7283:
   READ X:S=S+X:POKE Z,X:
   NEXT Z
20 IF S<>12457 THEN PRINT
   "ERROR EN LAS DATAS":END
40 SYS 7168:PRINT"[SHIFT+
   CLR/HOME]CORRECTO"
100 DATA 120,169,17,141,20,3
    ,169,28,141,21,3,88,169
    ,0,133,251,96
160 DATA 157,14,30,169,0,157
    ,4,150,232,136,208,239,
    76,1914
```

Tienes que copiar además las líneas del programa del **C-64** correspondientes al listado de la línea 30, así como las líneas 110 a 150, ya que son idénticas en ambos ordenadores.

Para poner de nuevo en marcha el reloj del **Vic**, teclea:

SYS 7168

DELTA
COMPUTERS, S. A.
Aribau, 15-6.º Dcho. 18
Tel. 253 97 91
08011 Barcelona

- PRECIOS CON IVA INCLUIDO
- GARANTIA OFICIAL
- PEDIDOS CONTRA REEMBOLSO + GASTOS DE ENVIO
- RAPIDEZ DE ENTREGA

COMMODORE 64 39.500
UNIDAD DE DISCO 1571 60.900
DATA RECORDER COMPATIBLE 4.900
LAPIZ OPTICO TOJAN CADMASTER 4.500
COMMODORE 128 59.900
IMPRESORA RITEMAN C+ 59.900
UNIDAD DE DISCO 1541 43.900
FUNDA ORDENADOR C-64 800

JOISTICK:

QUICKSHOOT I	1.395
QUICKSHOOT II	1.600
QUICKSHOOT IV	1.895
QUICKSHOOT V	1.695
QUICKSHOOT VII	1.600
QUICKSHOOT IX	2.295

:RITEMAN: news

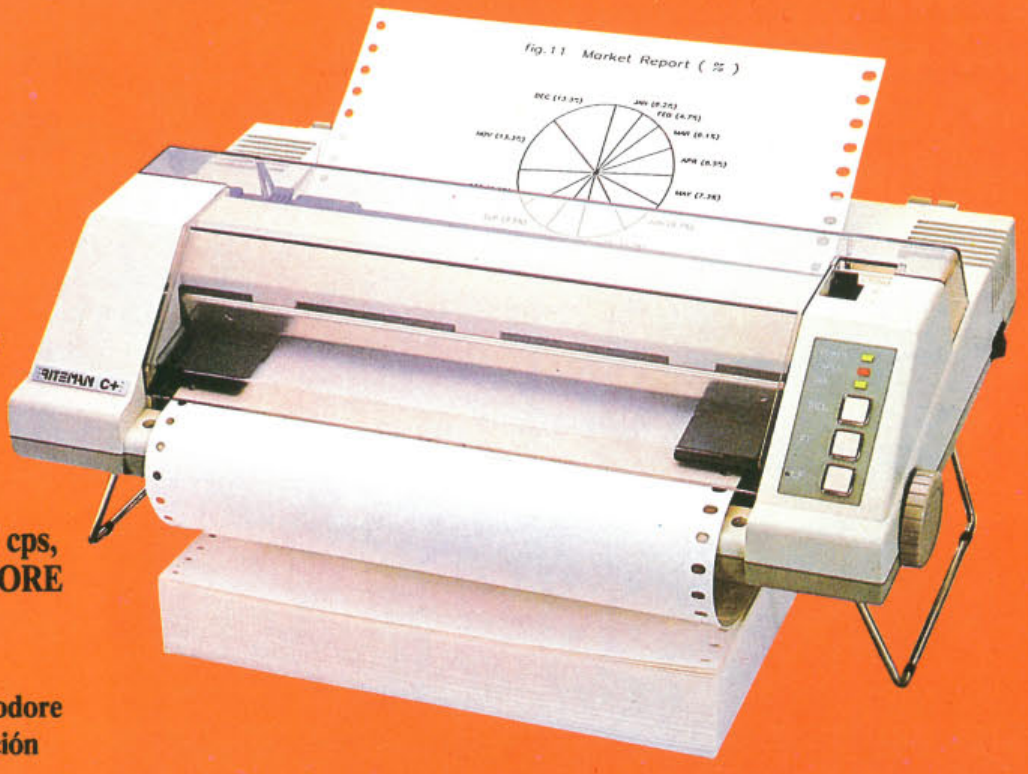
DATAMON REPRESENTACION EN ESPAÑA DE: :RITEMAN:

PROVENZA, 385-387
TEL. (93) 207 24 99*

TELEX 97791
08025 BARCELONA

IMPRESORA PARA SU COMMODORE
(óptima relación precio/prestaciones)

—Cabezal 9 agujas
—Doble operatividad
—Cinta autoretintada
—Tampón retintable
—Ausencia de rodillo
—No dobla el papel
—Elevadores inferiores
—Admite texto rígido
—Máximos tipos de escritura



Modelo SUPER C+, 120 cps, NLQ, ASCII y COMMODORE

• Conexión directa a Commodore (cable incl.) Tracción y fricción

LA IMPRESORA PARA COMMODORE, ASCII Y PC'S COMPATIBLES
(Máxima versatilidad/precio ajustado)

RITEMAN 10-C

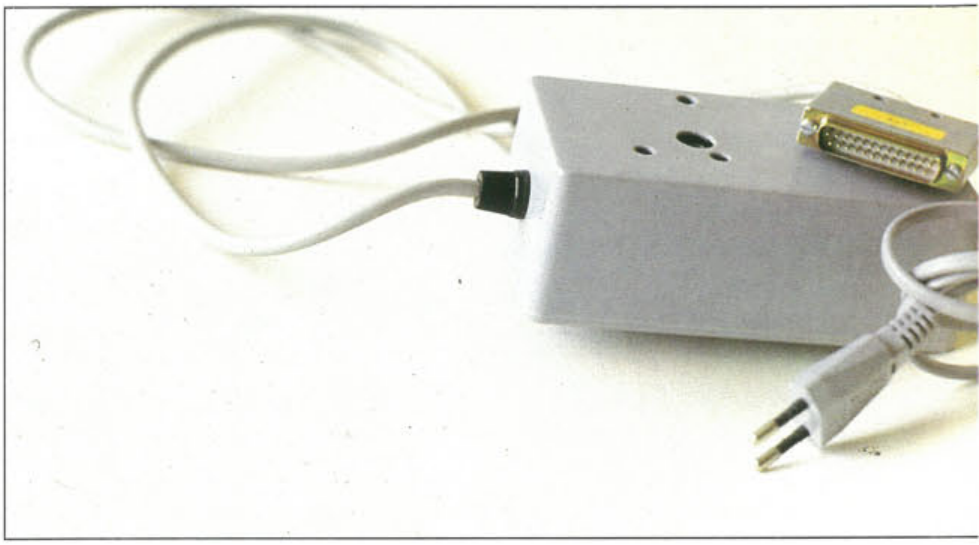
—140 cps, tracción y fricción
—Paralelo centronics/Commodore serie DIN
—Tablas ASCII y PC en Rom interna
—Tabla 100% Commodore y 8K RAM en módulo
—Interface Commodore exterior incluido
—RS 232-C opcional

NOTA: Para Aplicaciones en las que se necesite más velocidad, o mayor tamaño de carro, también pueden aplicarse nuestros interfaces externos a los modelos RITEMAN 10/II y RITEMAN 15.

Commodore es marca registrada de Commodore Business Machine, Inc.

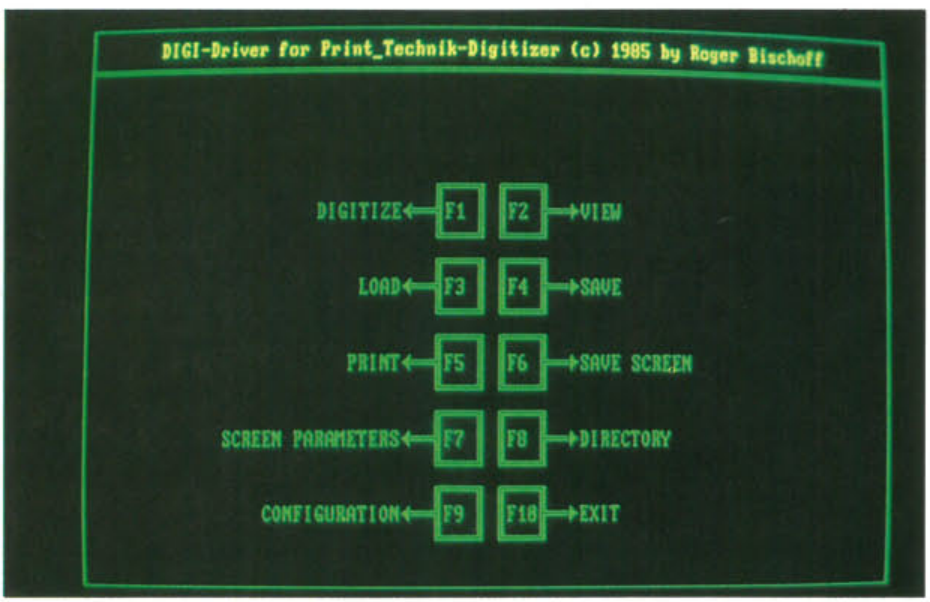
DIGITALIZADOR PARA PERSONAL COMPUTER

Hace un año que le dedicabamos un lugar en la revista a la descripción de un digitalizador de vídeo para el C-64. **Ahora le toca el turno a una versión con más prestaciones, también de** Print Technic, **destinada al** PC **de** Commodore **y a todos los compatibles con el estándar que marcó** IBM.



El punto de partida es la obligada fuente de señal de vídeo, tal como una cámara, un sistema de videoreproducción o incluso un ordenador.

El sistema precisa de los dos clásicos elementos bien definidos, *software* y *hardware*. La circuitería va incluida en el interior de una caja de plástico gris de dimensiones no muy grandes. De ella parten dos cables de conexión, el primero va directamente a la alimentación eléctrica del sector, mientras que el segundo termina en un conector que se acoplará al *port* paralelo del **PC** —utilizado habitualmente para la impresora. Por esto es imprescindible que la configuración del compatible incorpore dicha conexión. No todos los **PC** disponen de la tarjeta paralelo marcada por la norma **IBM**, este es el caso de algunos modelos de **Commodore**.



En la parte superior de la caja asoma un conector coaxial del mismo tipo que los habitualmente utilizados en los monitores. A través de él se inyecta la señal a digitalizar. Distribuidos en su entorno existen tres agujeros por los que puede apreciarse la ranura que permite girar el cursor de otros tantos potenciómetros ajustables. Sirven para ajustar los niveles de umbral de la señal y su recorrido es de 270 grados, aunque nos parèce que hubiera sido preferible emplear potenciómetros multivuelta, que permitirían mayores posibilidades de ajuste fino.

El *software* precisa del sistema operativo **PC-DOS** en versión 2.XX (aunque el **MS-DOS** funciona igualmente bien). Es preciso disponer, como parece obvio, de una tarjeta gráfica instalada en el ordenador. Las digitalizaciones se producen con una resolución de 65.536 *pixels* o, lo que es lo mismo, 256 por 256 puntos en 16 niveles de gris. En el sentido vertical ésta es mayor de la resolución de la pantalla, por lo que la imagen se puede desplazar hacia arriba y abajo mediante el cursor. Las dos teclas de cursor restantes se utilizan para alterar el color del contenido de la pantalla.

Una vez cargado el *software* específico desde el *diskette* aparece una menú en la pantalla, que reproduce esquemáticamente las diez teclas F1 a F10 y la función asignada a cada una. De este modo el manejo del digitalizador se convierte en algo extremadamente sencillo.

F1 corresponde a *Digitize* y al presionarla comienza el muestreo de la señal analógica de vídeo, para convertirla en números asociados con el nivel de gris de cada punto. La exploración de la imagen se produce siguiendo líneas verticales.

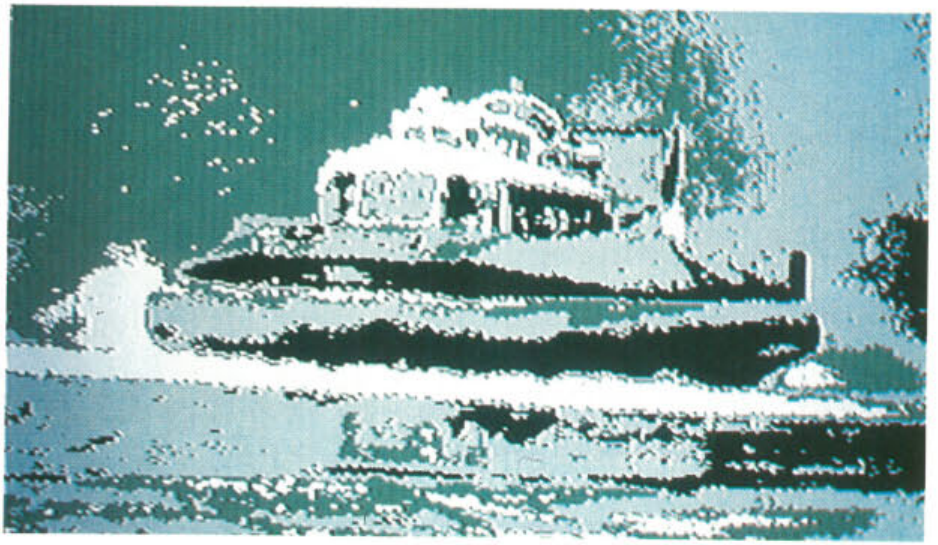
SAVE, estando por demás explicar su cometido, aunque sea asociado a imágenes.

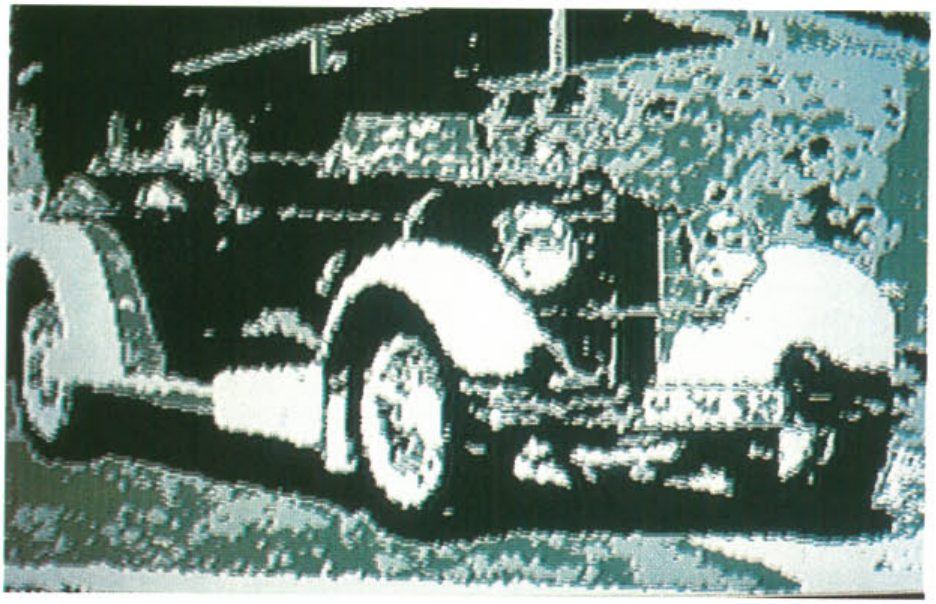
F5 permite volcar la imagen sobre el papel, a través de una impresora matricial. La tecla F9 puede haber sido empleada previamente para con-

almacena sólo los 256 por 256 puntos.

F7 permite definir los parámetros de la pantalla, tales como la elección de los 4 colores utilizados a otros tantos niveles de gris de los 16 disponibles.

F8 visualiza el directorio.





```
A>REM The IBM Personal Computer DOS
A>REM Version 2.00 (C)Copyright IBM Corp 1981,1982,198
A>
A>
A>dir

 Volume in drive A has no label
 Directory of  A:\

DIGIT    COM     14975  11-06-85   3:27p
DISP     COM      1636   7-31-25   7:40p
DISPEND  COM        10   1-01-80   9:44a
DISDEMO  BAT       103  10-16-85   9:10a
AUTO     SCN     16384   6-10-85   7:56p
NUT      SCN     16384   6-10-85   7:49p
AUTO2    SCN     16384   6-10-85   7:49p
LUFT     SCN     16384   6-10-85   7:50p
VIEMB    SCN     16384   6-10-85   7:54p
DIGCONF  DAT        82   1-01-80  12:23a
       10 File(s)     260096 bytes free

A>_
```

Mientras permanece una imagen en la pantalla, algunas teclas de función asumen un nuevo cometido. F1 permite digitalizar nuevamente una imagen, F3 invierte la imagen, F4 la rota 90 grados y F5 la transforma en su reflexión frente a un espejo.

F2 lleva la leyenda *View* (ver) y da paso a la imagen almacenada en el área de memoria utilizada por la pantalla gráfica. F3 y F4 son LOAD y

figurar los parametros de la impresora y el *interface*. Tanto la anchura como la altura de la imagen impresa pueden decidirse en esta opción con un amplio rango de variaciones. Igualmente se puede crear una seudocoloración, asociándole uno de 16 patrones de conformación de puntos a cada nivel de gris.

F6 permite almacenar pantallas en el *diskette*, por contrario a SAVE que

F10 se utiliza para salir del programa, devolviéndo el control al sistema operativo.

Como parece ser costumbre de este fabricante, el manual adolece de buen nivel de detalle. A pesar de haber desarrollado un dispositivo de gran utilidad en diversas aplicaciones. Algo más de información estamos seguros de que permitiría sacarle mayor provecho al sistema.

ENTENDIENDO LOS CODIGOS ASCII

- ¿QUE ES EL CODIGO ASCII?
- USO DE NUMEROS EN LUGAR DE CARACTERES
- PROGRAMA PARA IMPRIMIR MENSAJES CIFRADOS

El código ASCII empleado por los ordenadores tiene las suficientes semejanzas para permitirles literalmente «hablar» unos con otros. Este «lenguaje común» tiene también otros usos...

Cada tecla o combinación de teclas de tu ordenador está representada por un código electrónico único. Estos códigos se representan en BASIC por una serie de valores decimales que van desde 0 hasta 255. Por ejemplo, a la letra A le corresponde el valor decimal 65, a la B el 66, etcétera. Cuando tecleas letras o palabras cualesquiera, lo que el ordenador realmente almacena son estos códigos.

Los valores correspondientes a cada configuración de teclas no son los mismos en todos los ordenadores, pero afortunadamente existe al menos una cierta estandarización en la forma en que dichos valores se describen y utilizan.

Este conjunto de valores de designa por las siglas ASCII que significa Código Americano Normalizado para Intercambio de Información (*American Standard Code for Information Interchange*). Este código se elaboró para disponer de un medio por el que los ordenadores pudieran transferirse datos de unos a otros. Los **Commodore**, **Dragon**, **Spectrum** o **Tandy** utilizan este código, al menos en parte.

EL CODIGO ASCII

El conjunto completo de caracteres ASCII no es igual para todos los ordenadores. El mayor paralelismo se produce en el intervalo desde 33 hasta 90, que cubre todos los símbolos normales: signos de puntuación, números y letras mayúsculas. Hay una forma muy sencilla de encontrar el código ASCII de un carácter cualquiera; para ello no tienes más que teclear PRINT ASC(«X») y te aparecerá el código ASCII de «X» o de cualquier otro carácter en el que estés interesado. El siguiente programa te permite introducir lo caracteres con más facilidad.

Teclea para C-64 y Vic-20

```
10 PRINT"TECLEA UNA LETRA,
   NUMERO O CARACTER"
20 INPUT A$
30 PRINT "EL CODIGO ASCII
   DE ";A$;" ES ";ASC
   (A$)
40 GOTO 20
```

La función inversa de ASC es CHR$, que realiza la conversión del número de código a su correspondiente carácter. El programa que viene a continuación convierte en caracteres todos los códigos ASCII comprendidos entre el 33 y el 90:

Teclea para C-64 y Vic-20

```
10 PRINT "CODIGO ASCII",
   "CARACTER"
20 FOR N=33 TO 90
30 PRINT N;TAB(21);
   CHR$(N)
40 FOR D=1 TO 500:NEXT D
50 NEXT N
```

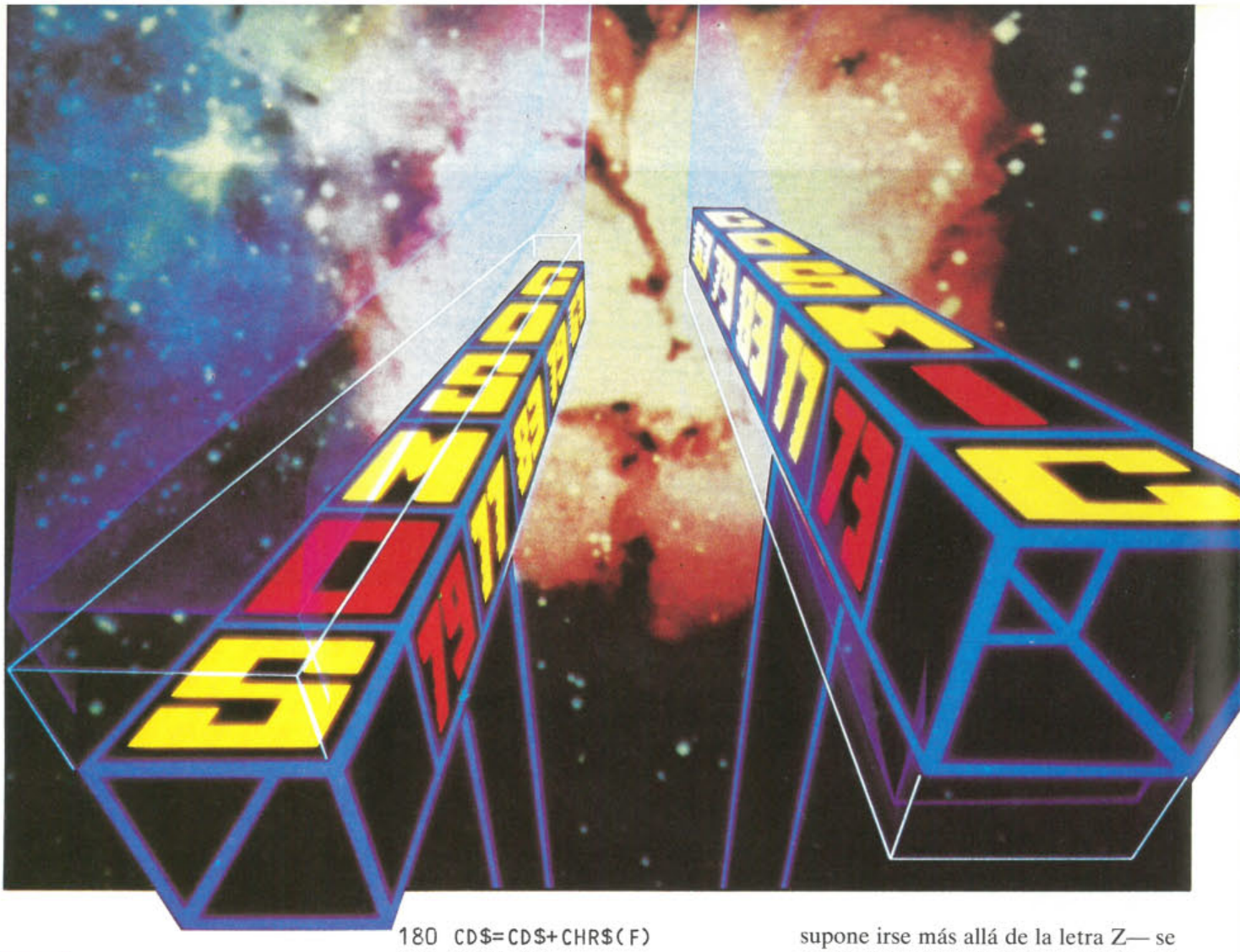
En el **Commodore 64** tienes que cambiar la representación en pantalla de mayúsculas/gráficos a mayúsculas/minúsculas, pulsando simultáneamente las teclas COMM y SHIFT. Esto se debe a que las minúsculas y los caracteres gráficos comparten los mismos códigos ASCII y lo que realmente tú ves depende del modo en que estés.

USO DEL CODIGO ASCII

La ventaja de utilizar un número en vez de un carácter, consiste en que puedes modificar matemáticamente dicho número de varias maneras. Si después de hacerlo imprimes el carácter del nuevo número, obtendrás una letra diferente. Esta es la base de muchos programas de escritura de códigos. Se trata de una utilidad de gran valor, ya que no es posible manipular directamente una letra aplicándole una operación matemática.

Los códigos más sencillos se limitan a añadir una cantidad constante a cada número, con lo que resulta de hecho una traslación de la letra en el alfabeto. Por ejemplo, la A se convierte en G, la B en H y así sucesivamente. Pero estos códigos resultan muy fáciles de romper; después de todo, no sería muy difícil escribir un programa de ordenador para probar cada una de las 26 combinaciones posibles, siendo muy fácil detectar de una ojeada cuál es la correcta.

Para que el programa tenga alguna eficacia, tiene que hacer cosas más complicadas con los números; el programa siguiente utiliza una palabra código de forma que cada letra del mensaje se altera de una manera diferente. Este tipo de codificación es muy difícil de romper, a menos que se sepa cuál es la palabra código.



Teclea para C-64 y Vic-20

```
10 PRINT"[SHIFT+CLR/HOME]"
30 INPUT "CUAL ES LA PALABRA
   CODIGO ";C$
40 PRINT "ESCRIBE EL MENSAJE
   :-"
50 INPUT MS$
60 CP=CP+1:IF CP>LEN(C$)
   THEN CP=1
70 PM=PM+1
80 IF PM>LEN(MS$) THEN 200
90 F$=MID$(MS$,PM,1)
100 IF F$<"A" OR F$>"Z" THEN
    150
110 F=ASC(F$)+ASC(MID$(C$,CP
    ,1))-65
120 IF F>90 THEN F=F-26
130 CD$=CD$+CHR$(F)
140 GOTO 60
150 IF F$<"0" OR F$>"9" THEN
    CD$=CD$+F$:GOTO 70
160 F=ASC(F$)+ASC(MID$(C$,CP
    ,1))-48
170 IF F>57 THEN F=F-10:GOTO
    170
180 CD$=CD$+CHR$(F)
190 GOTO 60
200 PRINT:PRINT "EL MENSAJE
    CODIFICADO ES:-"
210 PRINT:PRINT CD$
220 END
```

El programa te pide que introduzcas la palabra código y el mensaje, y a continuación imprime la versión cifrada. El mensaje ya está protegido y sólo lo podrá descifrar alguien que conozca la palabra código.

La forma en que trabaja el programa es muy sencilla y se parece mucho al programa de codificación simple, con la salvedad de que en vez de sumar una cantidad constante a cada letra, se le suma el código ASCII de la primera letra de la palabra código a la primera letra del mensaje, el de la segunda letra de la palabra código se suma a la segunda letra del mensaje, etcétera. Cuando se llega al final de la palabra código, el ciclo comienza de nuevo.

Si al sumar los números se obtiene un resultado mayor que 90 —lo que supone irse más allá de la letra Z— se restan 26 para mantenerse dentro del margen correspondiente a las letras del alfabeto. Los números reciben un tratamiento separado en las líneas 150 a 180, a fin de mantener sus códigos ASCII entre 48 y 57, que corresponden a los números 0 a 9.

El programa de descifrado es muy parecido al primero; la única diferencia consiste en que se han cambiado unos cuantos +s y −s. La numeración de las líneas es coherente con la del programa anterior, a fin de que puedas almacenar el programa combinado. Tienes además unas cuantas líneas adicionales que te permiten seleccionar la codificación o decodificación de un mensaje.

Teclea para C-64 y Vic-20

```
12 INPUT "(C)ODIFICAR O (D)
   ECODIFICAR ";A$
14 IF A$="D" THEN GOTO 400
16 IF A$<>"C" THEN GOTO 12
400 PRINT"[SHIFT+CLR/HOME]"
410 INPUT "CUAL ES LA
    PALABRA CODIGO ";C$
```

```
420 PRINT "ESCRIBE MENSAJE
    CODIFICADO:"
430 INPUT MS$
440 CP=CP+1:IF CP>LEN(C$)
    THEN CP=1
450 PM=PM+1
460 IF PM>LEN(MS$) THEN 580
470 F$=MID$(MS$,PM,1)
480 IF F$<"A" OR F$>"Z" THEN
    530
490 F=ASC(F$)-ASC(MID$(C$,CP
    ,1))+65
    500 IF F<65 THEN F=F+26
    510 CD$=CD$+CHR$(F)
    520 GOTO 440
    530 IF F$<"0" OR F$>"9"
    THEN CD$=CD$+F$:
    GOTO 450
540 F=ASC(F$)-ASC(MID$(C$,CP
    ,1))+48
550 IF F<48 THEN F=F+10:GOTO
    550
560 CD$=CD$+CHR$(F)
570 GOTO 440
580 PRINT "EL MENSAJE DECODI
    FICADO ES:-"
590 PRINT:PRINT CD$
600 END
```

Digamos de paso que si realmente quieres conseguir que tu código sea muy difícil de romper, siempre puedes volver a cifrar tu mensaje cifrado utilizando una segunda palabra código. Acuérdate de colocarlas en el orden correcto cuando descifres de nuevo el mensaje.

COMPARACIONES

Una de las principales aplicaciones de los valores del código ASCII es la comparación de cadenas de caracteres. Dichas comparaciones de cadenas se hacen carácter a carácter y de izquierda a derecha, hasta que se llega al final de la cadena. Supongamos por ejemplo que hay que comparar dos cadenas tales como «COSMOS» y «COSMIC». El ordenador primero compara las dos C, después las O, las S y las M. Seguidamente se compara la O de la primera cadena con el correspondiente carácter de la otra cadena, que es la I.

Lo que se compara no es un conjunto arbitrario de valores alfabéticos, sino el valor ASCII de cada carácter. El valor ASCII de la O es mayor que el de la I, por lo que en esta comparación «COSMOS» tiene un valor mayor que «COSMIC». Observa que no se utilizan las sumas de los valores ASCII de las letras para determinar cuál es la cadena que tiene un mayor valor. Cada letra se compara individualmente.

Aquí tienes como ejemplo algunas comparaciones que muestran posibles valores de A$ y B$, junto con los códigos ASCII de los caracteres que intervienen. Es importante comparar los valores uno a uno, el primer carácter de A$ con el primero de B$, siguiendo así en toda la extensión de la palabra más corta.

A$/ASCII		B$/ASCII		RELACION
ABC	65,66,67	ABC	65,66,67	A$ = B$
ABD	65,66,68	ABCD	65,66,67	A$ > B$
ABD	65,66,68	ABCD	65,66,67	A$ < > B$
ABC	65,66,67	Abc	65,97,98	A$ < B$
COSMI	67,79,83, 77,73	COSMO	67,79,83, 77,79	A$ < B$
$1	36,49	$1.0	36,49,46	A$ < B$

Como puedes observar en los ejemplos segundo y tercero, se puede escribir la relación de más de una forma. Fíjate también en el último ejemplo, en el que a igualdad de todo lo demás, se considera que es mayor la cadena de caracteres más larga.

COMPROBACION DE LAS ENTRADAS

La función ASC actúa únicamente sobre el primer carácter de la cadena. Así, la instrucción PRINT ASC(«USA») dará como resultado el número 85, que es el código ASCII de la letra «U». Este hecho resulta muy útil en la comprobación de las entradas que se suministran a los programas por medio del comando INPUT. Por ejemplo, en el anterior programa de escritura de códigos, la línea 12 te pedía que introdujeras la letra C para cifrar o la letra D para descifrar. A continuación, las dos líneas siguientes encaminaban el programa a la sección adecuada. Pero si quieres impedir que alguien escriba «cifrar» o «descifrar» *in extenso*, puedes volver a escribir así las líneas 14 y 16:

Teclea para C-64 y Vic-20

```
14 IF ASC(A$)=68 THEN GOTO
   400
16 IF ASC(A$)<>67 THEN GOTO
   12
```

CODIGOS DE CONTROL

Algunos códigos ASCII no van asociados a ningún carácter. Por ejemplo, cuando pulsas la tecla RETURN el ordenador almacena el valor 13, pero en lugar de imprimir un carácter en la pantalla el cursor se desplaza al principio de la siguiente línea o se lista el programa. Se dice que el código 13 es el código del retorno de carro, y en algunos casos el código de línea nueva. Teclea el comando PRINT «A»;CHR$(13);«B» y observarás que la «B» se imprime en una nueva línea directamente debajo de la «A».

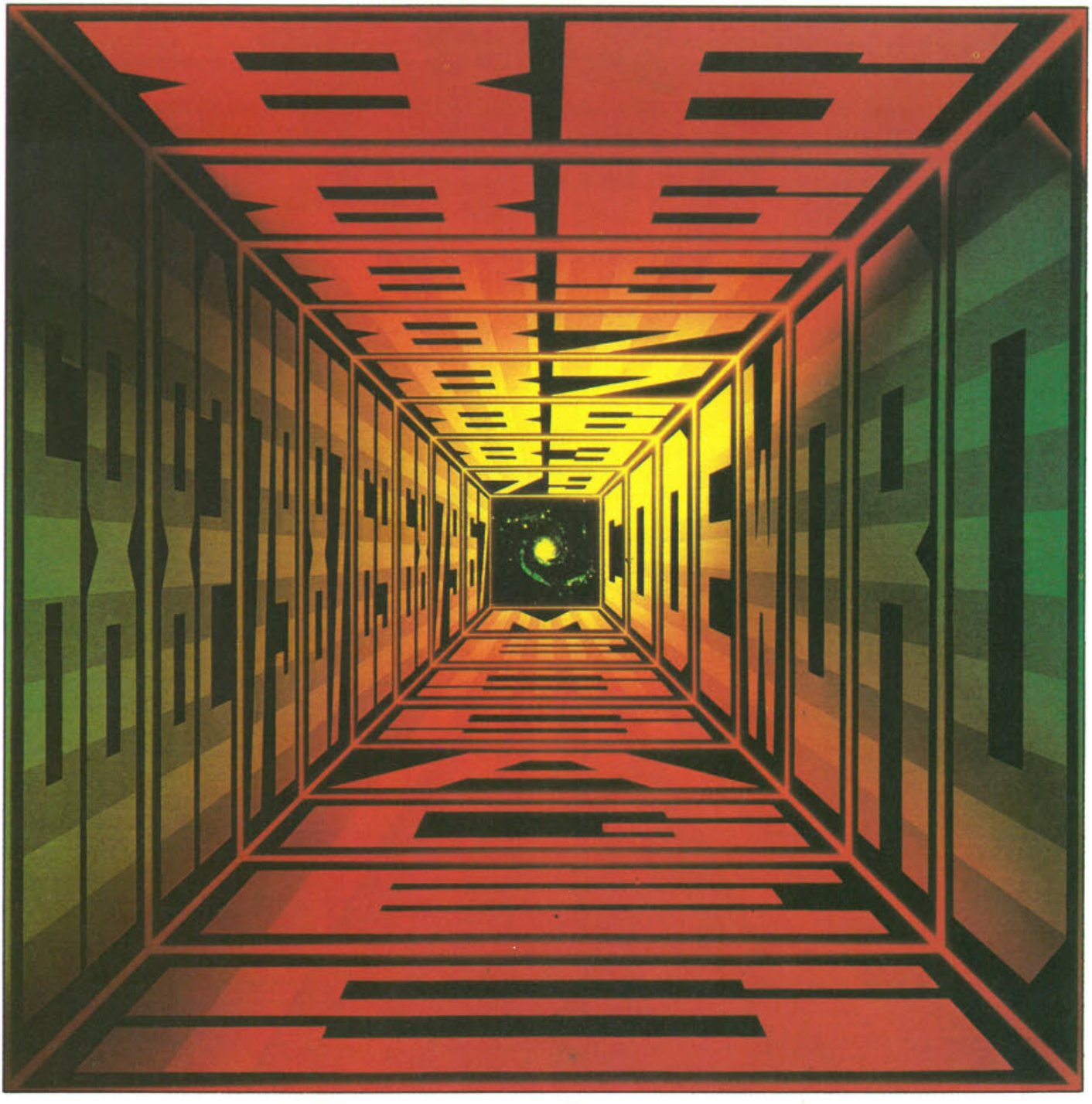
Aquí tienes la manera de encontrar los códigos ASCII de las teclas que no corresponden a caracteres:

Teclea para C-64 y Vic-20

```
5 GET A$:IF A$="" THEN 5
10 PRINT ASC(A$)
20 GOTO 10
```

Prueba a pulsar RETURN , DELETE , o cualquier otra tecla de tu ordenador. Puedes utilizar este método para detectar las pulsaciones de tecla en los programas de juegos. Por desgracia no puedes utilizar los valores de forma inversa, es decir, aunque el código 10 corresponde al movimiento del cursor hacia abajo, el comando PRINT CHR$(10) no hará que el cursor se mueva hacia abajo.

Unos ordenadores utilizan estos códigos más que otros. Hay muchos números disponibles: desde el 1 al 31 y unos cuantos por encima del 90, pero inicialmente fueron definidos en los días en que los ordenadores estaban conectados a impresoras, por lo que los códigos están relacionados fundamentalmente con el control de aquellas vetustas impresoras. Naturalmen-



te en la actualidad los ordenadores están diseñados para ser utilizados con un monitor de vídeo, por lo que la mayoría de los códigos ya no se aplican, si bien algunos de ellos se pueden utilizar en las impresoras modernas.

Los **Commodore** hacen un amplio uso de los valores de los códigos situados antes y después de las letras y los signos de puntuación. Entre los valores utilizados hasta el código 32 (espacio) se incluyen, entre otras cosas, varios de los colores disponibles, algunos de los controles del cursor y un cambio forzado a letras minúsculas. Uno de los valores que se utiliza con más frecuencia es CHR$(13), que corresponde a RETURN.

Los restantes colores y controles de cursor, además de los gráficos en ROM, se encuentran en los valores de código que siguen al 90 (correspondiente a la Z). Los códigos situados entre 191 y 255 se limitan a duplicar este último intervalo.

Se pueden encadenar códigos para obtener variables que lleven incorporados controles de cursor y de colores. Puedes comprobar el significado de los diferentes valores de los códigos con ayuda del apéndice de la Guía de Usuario. Por ejemplo, el código 14 hace que en la pantalla aparezcan letras minúsculas cuando se ejecuta el programa. Los nombres de las variables que se utilizan en dicho programa, dan una idea de las funciones realizadas por los valores de los códigos asignados. Observa cómo puede formarse una variable, tal como c4$ a partir de cd$ que había sido previamente definida por medio de CHR$(17). Como puedes ver, es posible realizar gran cantidad de manipulaciones sobre la pantalla.

LOS MEJORES DE INPUT COMMODORE

PUESTO	TITULO	PORCENTAJE
1.º	Commando	18,4 %
2.º	Green Beret	17,6 %
3.º	Skyfox	13,8 %
4.º	Saboteur	13,1 %
5.º	Hardball	10 %
6.º	Uridium	8,4 %
7.º	Rambo	6,5 %
8.º	Kung fu master	5,3 %
9.º	Fight night	3,8 %
10.º	World cup carnival	3,1 %
		100 %

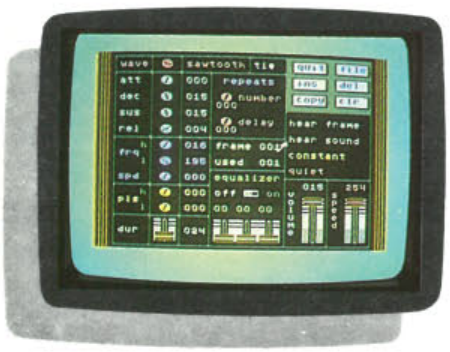
Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «Los Mejores de Input».

Septiembre de 1986.

EL HACEDOR DE JUEGOS

Puede que estés algo cansado de jugar con programas cuyas estrategias han sido diseñadas por otros para tí. Sin embargo, es preciso ser un programador experimentado para desarrollar un juego con un mínimo de atractivo. ¡Pero, *Voila* la solución! Games Maker **es un conjunto de herramientas de *software* que convierten el desarrollo de juegos personalizados en eso, un juego.**

Muchas veces habrás tenido una idea que juzgaste excelente para dar



paso a un juego. Después has puesto manos a la obra, viendo que la cosa no es tan simple, y seguro que te habrás hecho la siguiente reflexión: ¡Caramba, que difícil es programar! sobre todo cuando el código máquina está de por medio.

GameMaker ha sido desarrollado por **Garry Kitchen** para simplificar al máximo nuestro esfuerzo a la luz de una idea inicial. Este increible programador preparó hace algún tiempo una herramienta de similares características, aunque menos completa, llamada **Electronic Pencil** (El lápiz electrónico). De todas formas no es el único programa de estas características existente, **Epyx** dispone del **Programer's BASIC Toolkit**, que incorpora también nuevos comandos para el BASIC y que describiremos cuando sea lanzado aquí.

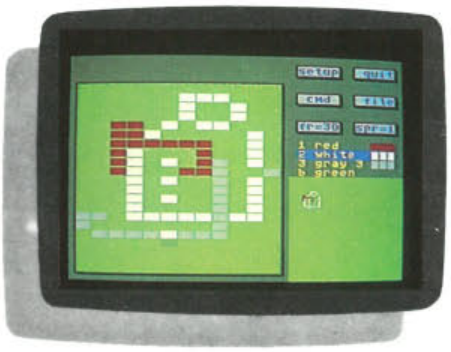
El **GameMaker** incorpora un particular lenguaje de uso específico, que viene a ser como manejar coloquialmente el idioma inglés. Conviene tener en cuenta que no le saca el mismo partido a un idioma una criatura que un avezado literato. Algo así ocurre con el nivel de experiencia que tenga un usuario de este paquete.

Se incluye asimismo una biblioteca de paisajes de fondo, sonidos, música y *sprites*.

La herramienta se compone de cinco bloques modulares, cada uno con un fin específico: **SpriteMaker**, **SoundMaker**, **SceneMaker**, **MusicMaker** y **Editor**.

Tal vez el que antes excita nuestra curiosidad sea el editor de *sprites*.

Aparecen dos ventanas con un formato muy en la línea de otros editores de *sprites*. La de mayor tamaño se utiliza para trabajar, mostrando nues-



tro diseño en grandes proporciones. En la menor se ve el resultado a escala real y en ella se pueden posicionar y animar los *sprites*. En un juego pueden aparecer hasta doscientos cincuenta y cinco *sprites*, si bien sólamente lo harán ocho a un tiempo. Para facilitar el manejo de los mismos, pueden ser bautizados en el programa mediante nombres alegóricos. Por ejemplo: *sprite* 2 is cazador.

Seguidamente, lo más adecuado puede ser la creación de un fondo que proporcione la atmósfera que deseamos para nuestro juego. Para ello podemos dibujar una escena propia, con ayuda de menú y *joystick*, o sacar alguno de los almacenados en el *diskette* original. Existen paisajes boscosos e incluso marinos. Partiendo de uno de ellos es factible retocarlo a conciencia para personalizar el diseño. El **SceneMaker** se asemeja bastante a otras

herramientas de dibujo, pueden trazarse líneas, cajas, círculos incluso rellenar áreas con un color. El *zoom* y su velocidad de manejo es algo soprendente por su carencia de parpadeo durante los desplazamientos.

Los efectos especiales de sonido son un elemento fundamental en los juegos. Las limitaciones impuestas por la tediosa labor de programación del *chip* SID del **Commodore 64** son superadas por el bloque **SoundMaker**, que nos permite ir probando distintas combinaciones a través de su menú hasta encontrar el sonido que nos satisfaga. La pantalla toma un aspecto que recuerda a una mesa de mezclas. Su manejo es sencillo y rápido.

La música se programa de manera similarmente fácil, pero esta vez disponemos incluso de tres pentagramas

en la pantalla, uno por voz del sintetizador interno. También aparece un teclado tipo piano, que facilita la identificación de las notas. En este módulo existe igualmente una biblioteca de temas musicales como punto de referencia.

Probablemente la mayor aportación del **GameMaker** es el **Editor**, que dispone de noventa comandos de programación seleccionables directamente mediante el *joystick*. Con él se aprovechan los elementos creados previamente con las otras cuatro herramientas, relacionándolos entre sí de la manera que juzguemos más conveniente.

En la tarea de edición la pantalla se divide en áreas. En la parte derecha se muestran ocho rectángulos en cuyo interior se leen otros tantos comandos maestros, que serán seleccionados para volver al almacenado en disco o guardar el trabajo, inicializar un disco, incluso «independizar» un disco de tal manera que un juego no necesite al **Game-Maker** para funcionar.

Una estrecha área rectangular situada en el borde superior de la pantalla proporciona mensajes, que tienen que ver con el manejo de los comandos más complejos o se asegura de que realmente queremos elegir una opción irreversible.

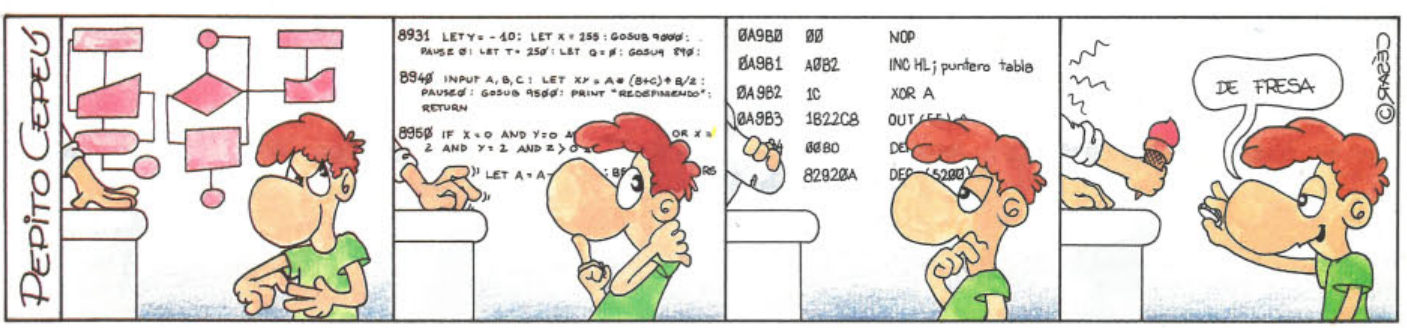
Debajo de ella existe otro área, llamada ventana de comandos y de tamaño definible, que visualiza los comandos disponibles para realizar la programación. Los comandos aparecen y se ocultan siguiendo un *scrolling* marcado por el *joystick*.

Finalmente, el área de programación es la que nos permite ir dándole forma a nuestro juego. Habremos de elegir los comandos directamente

de la ventana por la que asoman, situándolos en el orden adecuado, como si de un programa en BASIC o código máquina se tratara. Nuestras líneas se pueden volcar a impresora o ejecutar para comprobar su efecto.

Volvemos a incidir en que algo muy

a tener en cuenta es la extremada sencillez de manejo del **GameMaker**. Todas las opciones y posibilidades se eligen directamente con el movimiento de *joystick*. Estamos seguros de que un programador suficientemente experimentado con el **GameMaker** puede obtener un juego con buen aspecto comercial. Tanto es así que la propia **Activision** ha lanzado el reto y está dispuesta a patrocinar el juego ganador de un concurso entre los usuarios de este conjunto de herramientas.



TIEMPO PRESTADO

Todo comienza como en las historias de detectives. Una llamada telefónica te pone sobre aviso de un asesinato: El tuyo. Alguien pretende deshacerse de un incómodo investigador privado. Sin embargo no se te aclara quien y los motivos. Por otro lado, tu ex-esposa **Rita** tiene un amante rencoroso...

Borrowed Time es un juego de aventuras interactivas ilustrado, en el que debes teclear las órdenes, tanto para desplazarte como para saber en qué lugar de la investigación te hallas. Puedes coger algo, examinar, ir en una dirección, dejar objetos, etc.

El programa dispone de un elaborado comprobador de sintáxis, que te permite emplear un inglés casi coloquial (el juego no está traducido, por lo que puede ser un buen elemento de práctica del idioma).

Debes ser observador y comprobar en las imágenes fijas, que van sucediéndose, algunos de los elementos de los que te habla el ordenador. También recogerás información de tus amigos y enemigos.

Este juego demuestra tener personalidad propia, ofreciendo un interés muy superior a los demás de su clase. Se nota que detrás hay muchisimas horas de programación.

La versión que comprobamos venía en disco y sorprende ver como muchas situaciones son rápidamente comprobadas en él, de forma casi inmediata, por lo que en realidad utiliza mucha más memoria de la disponible internamente en el ordenador.

Está contemplada la opción de almacenar el juego en cualquier fase que hayas alcanzado.

En principio debes jugar con el teclado, pero existe una interesante opción de juego con el *joystick* y un cursor con forma de flecha.

Borrowed Time nos sorprendió favorablemente la primera vez que lo vimos, pero más aún cuando nos enfrentamos a él durante algunas horas, a pesar de ser muertos inmumerables veces en las primeras partidas.

Además tiene moraleja, no siempre una pistola lo es todo, a veces vale más la deducción.

DATOS GENERALES

TITULO Borrowed Time

FABRICANTE Activision

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Aventura con graficos

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9
INTERES	9,5
GRAFICOS	8
COLOR	8
SONIDO	8
TOTAL	42,5

★★★★★★★★★★★★★★★★★★★★★★★★★

EL PASTOR Y SUS OVEJAS

El escenario en el que transcurre la acción del juego son las llanuras del planeta **Andrómeda**. Hasta allí han llegado el jugador y su droide, versiones intergalácticas de un pastor de ovejas y de su perro, con el objetivo de capturar el mayor número posible de ramboides.

Son estas una rara especie de ovejas

DATOS GENERALES

TITULO One Man Droid

FABRICANTE Mastertronic

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Arcade

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	**7,5**
INTERES	**7**
GRAFICOS	**8**
COLOR	**7,5**
SONIDO	**7**
TOTAL	**37**

salvajes, que habitan el planeta. Curiosamente estas ramboides tienen un período de vida activa de únicamente 20 minutos. El jugador tiene el tiempo justo y debe trabajar deprisa para capturar el mínimo de 4 ramboides que le permiten pasar a otra de las veinte

¡PROGRAMADORES¡

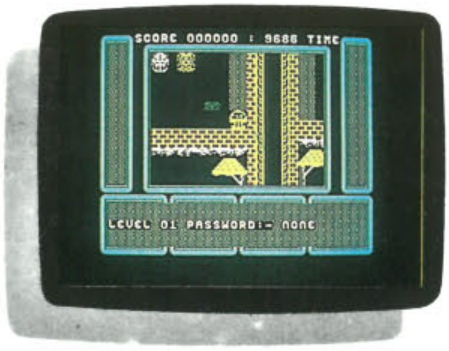
ERBE

OS PROGRAMA UN ATRACTIVO FUTURO

1	Ponte en contacto con nosotros
2	Desarrolla tus ideas con nuestro equipo de expertos
3	Ve como tu proyecto sale al mercado con una buena presentación y una gran promoción.
4	Ve como sube en las listas
5	Ponte cómodo
6	¡Elige tu Porsche!
7	Reserva tus vacaciones en Hawai
8	Busca un asesor para hacer tus declaraciones de Hacienda

Ahora en serio... cuando se trata de desarrollar y promocionar Software para Spectrum, Commodore, Amstrad o MSX no hay nadie mejor ni con mayor experiencia que ERBE. Tanto si tienes algún juego acabado, ideas en desarrollo o simplemente gran habilidad para programar o crear gráficos, ponte en contacto hoy mismo con ERBE.
Podría ser tu primer paso hacia un futuro muy atractivo.

ERBE SANTA ENGRACIA, 17 - 28010 MADRID
TEL. (91) 447 34 10



cavernas del planeta. El droide es una ayuda inestimable para el jugador . Con sus tres modalidades de funcionamiento, seleccionables a través de un cuadro que aparece en la parte inferior de la pantalla, es capaz de manejar a las ovejas a gran velocidad. Las tres modalidades del droide son las de vuelo, excavación y taladro de túneles. Cada una de ellas tiene sus ventajas y sus inconvenientes. Gran parte del éxito estriba en saber cual es la modalidad que conviene utilizar en cada momento.

El juego, que pertenece a la gama de bajo precio de **Mastertronic**, es sobre todo entretenido. Aspectos como los gráficos, el movimiento y el sonido, sin ser nada excepcional, resultan adecuados y contribuyen a un resultado global excelente.
El programa ofrece una opción interesante que consiste en poder retomar el juego en la pantalla en que lo dejamos. Para ello sólo hay que recordar determinada palabra clave que el programa nos da.

★★★★★★★★★★★★★★★★★★★★★★★★★★

EL GRAN INVERSOR

Invierte y Gana nos ofrece la posibilidad de entrar en el apasionante y misterioso mundo de las finanzas. Es un juego de simulación de la actividad de una bolsa de valores imaginaria.
En ella tendrán que desenvolverse hasta seis jugadores, vigilando el estado de las cotizaciones y decidiendo la compra o venta de acciones, siempre con el objetivo final de maximizar el beneficio.
La presentación del juego es bastante clásica, se controla desde un menú de ocho opciones. A partir de él conoceremos las cotizaciones, el estado de nuestra cartera de valores, la compra y venta de acciones y la realizacion de operaciones bancarias, tales como la solicitud o devolución de préstamos.
Cuando elegimos una opción, aparece en la pantalla un cuadro con la información correspondiente. A continuación el jugador es preguntado por el paso a dar.
La bolsa de valores incluye venticinco sociedades, repartidas en cinco sectores: Bancos, eléctricas, seguros, petróleos e industria.
Cada una pone a la venta dos mil acciones con un valor unitario de cien enteros, igual a quinientas unidades monetarias.
El capital inicial de cada jugador es de quinientas mil unidades monetarias.
Durante el juego, las cotizaciones fluctuan en función de las compras o ventas que realizan los jugadores, aunque también el programa genera factores aleatorios. Este tipo de informaciones aprarecen en *flashes* en la pantalla.
Las partidas pueden ser bastante largas, echándose de menos un opción que permita almacenar el resultado parcial del juego en cinta.

DATOS GENERALES

TITULO Invierte y Gana

FABRICANTE Dimension New

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Juego de bolsa

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	8
GRAFICOS	7
COLOR	7
SONIDO	6
TOTAL	36

★★★★★★★★★★★★★★★★★★★★★★★★★★

PROPULSION

Pilotas una nave que parece flotar libremente en la inmensidad del espacio. Estas a merced de la propulsión que proporciona el motor principal.
Thrust no es un juego espectacular, pero es inmensamente agradable de manejar. Puedes hacer que la nave rote en torno a su propio eje en los sentidos igual y contrario que las agujas del reloj.
Sin embargo el juego no consiste únicamente en darte tranquilos paseos espaciales, sino en conseguir las fuentes de energía que

permitirán a las potentes naves de la resistencia actuar contra el tiránico **Imperio Intergaláctico**.
Vas avanzando por los siete niveles existentes, en la medida en que destruyes las estaciones de servicio a las que previamente has desposeido de los cristales energéticos.
Cada estación está protegida por una batería que dispara contra ti y que se alimenta por medio de una central nuclear. Puedes desactivarlas durante un rato si haces blanco en ella. A mayor cantidad de blancos más tiempo disponible, pero si te pasas se producirá una explosión atómica que hará desaparecer el planeta en diez segundos.
Es curioso, pero existen otros mundos cuya gravedad es inversa, aumenta con la distancia que te separa del suelo.
El control no se realiza con *joystick*, sino que con siete teclas. La dificultad aumenta cuando quieres hacer muchas cosas a un tiempo. A pesar de la simple apariencia inicial del juego, hay que trabajarlo para llegar a un determinado nivel de dominio.
Es muy fácil estrellarse contra el

suelo si no se dominan adecuadamente los propulsores y la cantidad de movimiento que se les hace aplicar. Hay que controlar con mucho cuidado el ángulo de inclinación del morro de la nave.
Aunque los dibujos responden al tipo esquemático, el juego dispone de todos los elementos de adictividad que han hecho triunfar a muchos otros.

DATOS GENERALES

TITULO Thrust

FABRICANTE Firebird

ORDENADOR Commodore 64

TEMA DEL PROGRAMA Arcade

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	7
GRAFICOS	7
COLOR	6,5
SONIDO	6
TOTAL	34,5

★★★★★★★★★★★★★★★★★★★★★★★★★

BOMBJACK

Este programa es la réplica del popular juego del mismo nombre que conoceréis de los bares.
El protagonista del juego es una especie de minisuperman con capa y traje de colorines.
Partes de la conocida pantalla con la esfinge al fondo, por la que deber ir recogiendo las bombas dispersas por doquier en grupos de plataformas. Hay que evitar a las criaturas diabólicas y el tenáz pajarraco que te persigue. Pueden derribarte al mínimo roce.
Existen cinco pantallas diferentes, cada una más difícil que la anterior. Aparte de la esfinge, existe un templo griego, un castillo gótico, rascacielos y edificios.
El personaje, **Jack**, es controlado directamente mediante el *joystick* y puede realizar fantásticos saltos que son casi vuelos.
Existen unos discos giratorios marcados con letras, que acuden en tu ayuda. Los marcados

con la P congelan a los monstruos durante un ratito. Los discos E dan más cantidad de vida y los B dan bonificaciones, siempre que logres alcanzarlos.
La velocidad del desplazamiento y la calidad de los gráficos son todo lo buenos que podría esperarse de **Elite**. Aún tratándose de un juego de apariencia sencilla, es enormemente adictivo.

DATOS GENERALES

TITULO Bombjack

FABRICANTE Elite

ORDENADOR Commodore 64

TEMA DEL PROGRAMA Arcade

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7,5
INTERES	6,5
GRAFICOS	9
COLOR	9
SONIDO	9,5
TOTAL	41,5

★★★★★★★★★★★★★★★★★★★★★★★★★★

SOFTACTUALIDAD

SAUCER ATTACK II

La firma **Sach** nos ha hecho esperar por la segunda parte de **Saucer Attack**.
Encarnas a un piloto de nave temporal, y como tal deberás empezar por los origenes de la humanidad. Nuestro planeta está poblado de dinosaurios de color rojo sangre, cuyo nivel de detalle gráfico te hará estremecer. La animación es digna de la primera versión, pero actualizada.

★★★★★★★★★★★★

TWO ON TWO

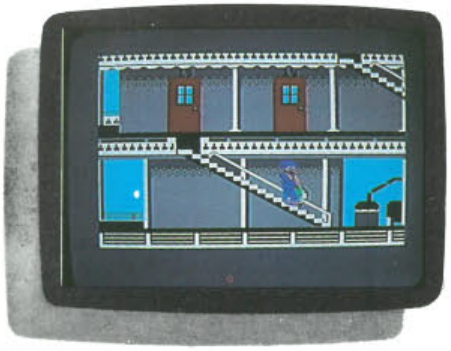
Este es un sensacional juego de baloncesto producido por **Activision**. Lo novedoso del programa es el campo abierto, por el hecho de que sean dos los jugadores en cada bando.
Son posibles pases largos, fuertes y el sonido los dota de un realismo sorprendente. El juego incluye a muchos de los jugadores de la liga profesional de baloncesto americano

de la NBA. Quizás lo más atractivo sea la posibilidad de jugar uno contra otro teniendo cada uno por compañero al ordenador.

★★★★★★★★★★★★

ASESINATO EN EL MISISSIPI

Otra de las novedades de **Activision** es un asesinato en el Misissipi en la época de los barcos de vapor de **Marc Twain**.
Un juego que es una muestra clara de la tendencia que estan siguiendo los juegos americanos conversacionales, que tratan de



simplificar la relación entre el jugador y el ordenador.
Aquí todo está ya resuelto y se trata simplemente de elegir entre varias opciones.
Un barco de lujo remonta lentamente el río. Los personajes que lo pueblan son de lo más variopinto: jugadores profesionales, esclavos evadidos, insurrectos, indios en busca de bronca...

★★★★★★★★★★★★

PING PONG

Muchos no quedaron demasiado contentos con las limitadas posibilidades del **Ping Pong** de **Konami**, una réplica del juego oriental no ha tardado en aparecer en versión mucho más compleja.
Todos los efectos están incluidos, uno y dos jugadores, varios tipos de estilos de saque, de perspectivas, de velocidades y bote de la bola, gráficamente incluso la proporción entre la bola, la raqueta y la mesa es más realista que en el juego que fué primicia.

★★★★★★★★★★★★

CONFLICT IN VIETNAM

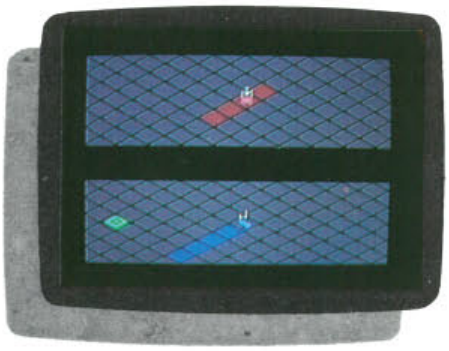
Tras las azañas de **Rambo** en Vietnam parece que esta zona del globo ha cobrado un nuevo interés.
Conflict in Vietnam es un juego de la casa **Microprose** de quienes se dice y se comenta que nunca han realizado un esfuerzo tan fenomenal para crear un juego de estrategia. La intención

es lograr que Vietnam acabe de otro modo.

★★★★★★★★★★★★

TRON CONSTRUCTION SET

Todos recordaréis un juego de una brocha de pintar que fué famoso hace un par de años, en aquel el juego estaba planteado con una sola perspectiva, desde arriba.
Este se plantea como una mezcla de **Pitsop II** (dos tableros para el caso en que juegan dos personas), punto de vista tridimensional, e incluso



constructor de pantallas muy fácil de manejo.

★★★★★★★★★★★★

PIRATES

¡Vaya piratas con cara de malos que han aparecido en los mares del **C-64**! Y para colmo de la mala suerte se van a enfrentar uno contra otro. La batalla va a empezar, los dos barcos

están unidos por una maraña de cuerdas. Desde las diferentes plantas del barco van saliendo cantidades de piratas, que han de librar batalla entre las cuerdas. La caracterización de los personajes le añade enormes cantidades de simpatía.

SI QUIERES VER PUBLICADOS TUS SPRITES

Con toda seguridad diseñaréis magníficos *sprites* perfectamente utilizables en juegos. Enviándonos las listas con los DATA que los dibujan, publicaremos aquellos que puedan ser empleados por otros usuarios para desarrollar sus programas.
Escribir en el sobre: SPRITES

EL ZOCO

Intercambio últimos programas por programas de juegos o gestión. Escribid a:

Jesús Blanco Garcillan
Calasparra 23
28033 Madrid

Vendo carcasa Atari, junto con cartuchos y mandos compatibles con el CBM 64. También intercambio programas para este.

Francisco Martínez Castillo
Perez Soto, 4
Tel. 64 45 46
38009 S. C. de Tenerife

Vendo 3 juegos para CBM 64: Countdown to Meltdawn (3-D), Hunchback II, y Solo Flight (3-D). Estupendos gráficos. 1500 ptas. cada uno.

Pablo
Tel. 202 22 06
Madrid

Vendo Commodore Vic 20, Datassette C2N, cartucho Super-Expander, cartucho de juego Vic-Avenger, Joystick Quick-Shot II, cinta de juego Skramble (original), cinta de juego Golf (original), libro «Introducción al lenguaje BASIC: I parte», libro «Programas comentados de BASIC Básico» y revistas Commodore, todo junto por 45.000 ptas. Interesados diríjanse a:

Carlos Galán
Mestre Millet, B, 2.ª
Tel. (93) 662 50 90
Gavá (Barcelona)

Vendo Commodore 64 (equipo completo con embalaje de fábrica) + 35 programas (entre los cuales están The Dambusters, One on One, Boulder Dash, Hunchback, Pitstop II, Turbo (cassette), Procesador de Texto, etc.) + 2 libros («64 Consejos y Trucos» de Data Becker y «Como programar su Commodore 64» de ediciones Paraninfo). Todo en inmejorable estado por 40.000 ptas. Preguntar por:

Jesús Frias Reyes
Arroyo, 3 pº 4º D
Tel. (91) 215 83 55 de 8 a 11 de la noche
28029 Madrid

Vendo cartucho rádar Rat Race para C-64 ó 16. También curso de Basic Parte I en cassette y con instrucciones. Las dos cosas por 2.500 ó por Simon's Basic con instrucciones.

Xavier Serrano
Tel. 317 84 13 (6h.- 10h.)
Barcelona

Cambio juegos para Commodore 64 (tengo One on One, Falcon Patrol, Zaxxon, Pitfall, Solo Flight, Decathlon, Match Point, Pistop II, Boulder Dash, Saucer Atack, Hero, Cauldron, etc). Escribir a:

Tomás Fuentes Méndez
Avda. Constitución, 63
Tel. (968) 59 03 57
Mazarrón (Murcia)

Vendo video-juego Philips video pac G7000 con 5 cartuchos, 10.000 ptas.

Tel. 273 40 79
Madrid

Vendo Fast Turbo Menue de Astoc Data por el módico precio de 4.500 ptas (precio original 8.500) Regalaría algún juego (Falcon Patrol, Zaxxon, Impossible Mission, etc.)

Juan Carlos González Muñoz
Duplex 14, Los Cantos
Tel. (955) 59 11 59
Riotinto (Huelva)

Cambio órgano polifónico Yamaha PS 40 comprado en el verano del 85 (perfecto estado), por Commodore 64 con cassette. O bien lo vendo por 35.000 ptas. Con el órgano incluyo maleta y alimentador.

Diego Martín
Cerrada, 2, 6º I
Tel. (983) 26 32 01
Valladolid

Vendo C-64, Datassette, Joystick Quick Shot II, 3 meses de uso, con garantía, regalo 60 juegos y programas de utilidad, Mission Impossible, Simon's Basic, Simulador Spectrum, Ghostbuster, One on One. Regalo lote de revistas. Todo por 55.000 ptas. Más de 10.000 ptas en juegos.

Juan Chacón
Tel. (93) 372 40 97

Cambio programas para C-64 de la categoría de Blackwyche, Summer Games I-II, Zorro. También cambiaría por calculadoras programables, ordenadores de bolsillo; cantidad a convenir. Interesados escribir a:

Andrés Ramos González
Eras de San Isidro 7, 1ºC
Alcalá de Henares (Madrid)

Cambio buenísimos programas por: impresora, unidad de disco, joysticks o monitor color que estén en buen estado de uso. Los programas son del tipo Winter Games, Karateka, Rambo, Kawasaky Rythm Rocker, etc... Cantidad a convenir. Interesados escribir a:

José Alberto Críado
Murillo, 3, 1ºB
Alcalá de Henares (Madrid)

Por cambio de equipo vendo lo siguiente: Procesador de Textos (4.000 ptas.), Hoja electrónica Calc Result Avanced (cartucho) (18.000), Ampliación al Basic Simon's Basic, Magic Desc (cartucho) (5.000), Music Composer (cartucho) (5.000), Basic Adventures in Space (part. 1) The Alien Planet (Dr. Watson Series) (cassette) (5.000), Iniciación al Basic (1ª parte) (2.000), Datassette Commodore (6.000), Commodore 64K (35.000), Impresora aplicable al CMB-64 Brother Ep-44 (65.000). Todo junto lo vendo por 120.000 ptas.

Manolo de Luque Muntaner
Pablo Piferrer, 15, 5ºC
Tel. (971) 23 19 20
07011 Palma de Mallorca
Baleares

Intercambio programas para C-64. Interesados escribir a:

José María Gómez
Domingo Beltrán, 1-B, 1ºdcha
01008 Vitoria

Intercambio todo tipo de juegos y programas de aplicaciones para CBM 64.

Carlos A. Pérez
Tel. 751 68 07
Barcelona

Usuario de CBM-64, disco e impresora Seikosha SP-800, desea intercambiar experiencias y programas.

J.A. Larumbe
Rucabado, 14, 4ºE
Castro Urdiales (Cantabria)

Regalo Commodore 64 (manual, fuente de alimentaicón, embalaje de fábrica) + 2 libros (64 Con. y Tru.(Data Becker) y «Como programar su C-64. I» (Paraninfo)) + 50 programas (entre ellos The Dambusters, Solo Flight, Decathlon, Hunchback, Turbo (cassette), Proc. de Texto). Todo en estado impecable por 30.000 ptas.

Jesús Frias Reyes
Arroyo, 3, 4ºD
Tel. (91) 215 83 55 llamar de 7 p.m. a 11 p.m.
28029 Madrid

AMIGA by Commodore

TARDAREMOS EN VER ALGO PARECIDO.

En informática ya es difícil sorprender. Pues bien, Commodore lo ha conseguido con Amiga.

Más que una nueva generación de ordenadores, el Amiga de Commodore representa un nuevo concepto. Un ordenador que ofrece una serie de posibilidades y abre unos caminos que hasta hoy eran impensables.

Entre las novedades de este ordenador profesional destacan: la posibilidad de trabajar con un procesador de 32 bits y de 3 coprocesadores específicos, actuando los 4 al tiempo.

Además, Amiga dispone de pantallas y ventanas configurables con una resolución de 640 x 400 puntos escogiendo entre una paleta de 4.096 colores.

Trabaja con 4 canales independientes polifónicos con voz masculina y femenina, y por si fuera poco, Amiga de Commodore es el único ordenador multitarea que puede efectuar varios trabajos simultáneamente.

El precio también es importante. Por 330.000 ptas. ex. IVA, el Amiga incluye:

- Unidad central con 768 K RAM.
- Monitor de alta resolución en color y sonido.
- Unidad de discos de 880 K.
- Teclado profesional y ratón.

Por mucho que avance la investigación en el campo de los ordenadores, tardaremos en ver algo parecido.

© Bassat, Ogilvy & Mather

commodore

Imagina siempre lo mejor.

Microelectrónica y Control, S.A. *Unico representante de Commodore en España.*
c/ Valencia, 49-53 Tel.(93) 325 57 02 08015 Barcelona - c/ Ardemans, 8 Tel. (91) 246 37 36 28028 Madrid

EL ZOCO

Busco adaptador a TV para Vic 20; asimismo busco cualquier tipo de cartuchos o ampliaciones para Vic 20. Intercambio programas de todo tipo para C-64.

Herminio Manzano Bojados
Avda. José Molins, 9 - 4.1.
Tel. 249 50 03
L'Hospitalet (Barcelona)

Vendo por equivocación en la compra, ordenador HIT BIT MSX, con la grabadora, más diversos juegos y libros todo junto por 40.000 ptas. o por separado. Todo a estrenar.

Javi
Tel. (981) 28 71 64

Vendo dos Commodore 64 + dos Datassette + una unidad de discos (1541) + una impresora Seikosha (G.P. 500 VC) + un Vic-Switch, en perfecto estado y a un precio de fábula. Se vende todo o si se prefiere por separado, se incluyen juegos (cassette o disco) y manuales y libros de utilidad.

Antonio Andreu Del Aguila
Pelayo, 4
Tel. (93) 664 16 04
Castelldefels (Barcelona)

Cambio muy buenos juegos para el C-64 en cinta. Todos son originales, aprovechaos amigos Commodoreros de esta única ocasión. Los interesados escribir a:

Francisco Javier Calero Fernández
Simancas, 2, 1°A
Melilla

Vendo Vic 20, cartucho super expander, guía de referencia del usuario, 1ª parte curso de Basic más listados, todo por 20.000 ptas.

Angel D. Basildo Alvarez
Hnos. de Pablo, 42, bajo A-23
Tel. 267 25 67 (llamar por las mañanas)
28027 Madrid

Intercambio, y me gustaria encontrar programas al código máquina. No tengo ni idea de código máquina «para Vic 20» e intercambio juegos.

Miguel Angel Romero
Mascatarro bl. B 1° 4ª
Villafranca del Penedés
Barcelona

Deseo cambiar juegos para Commodore 64 en disco o cinta.

Ramón Torres Pérez
Sepúlveda, 171, 2° 1ª
Tel. 254 69 71
08011 Barcelona

Vendo C-64, datassette C2N, y joystick por 45.000 ptas. Además regalo los juegos y utilidades como Koala Pad, Pitfall II, One on One, etc. También vendería sólo el C-64 por 40.000 ptas. 6 meses de uso. Llamar a:

Quique
Tel. (93) 223 13 57
Llamar de lunes a jueves a partir de las 16 horas

Hemos creado un club de Commodore 64. Intercambiamos juegos. Interesados escribir a:

Alberto Cases Santafé
Avda. Galán Bergua, 39, 3°A
Alcañiz (Teruel)

Vendo videojuegos T.R.Q. Consola 10.000 ptas. 6 Cartuchos 5.000 ptas. Todo por 10.000 ptas. Dirigirse a:

C. Urtazun
García Jimenez, 5, 1° izda.
31002 Pamplona

Me gustaría intercambiar juegos de Commodore 64. Poseo parte de las últimas novedades. Mandar lista.

Guillermo García
Pérez Galdós, 20, 3° Ext. Izda.
Tel. 443 93 22
48010 Bilbao

Intercambio programas y experiencias sobre C/64 y C/128.

Alfonso Franco Benítez
Bda. Los Naranjos, 20, 3° A
Tel. 32 05 87 (a partir de las 6 de la tarde)
Jerez de la Frontera (Cádiz)

Vendo Vic 20 + ampliación 8K + 2 cintas de iniciación al Basic + 2 libros del mismo tema para Vic 20. Todo ello por 30.000 ptas.

Aitor Pescador Medrano
Pablo de Alzola, 4, 6°C
Tel. (94) 443 82 79
48012 Bilbao

Estudiante de ingeniería vende C-64 por cambio a ordenador mayor, junto con la unidad de cassette, 39 juegos de la categoría de Ghostbusters, Solo Flight, Soccer 64, One on One... libros y manuales, todo en perfecto estado por 55.000 ptas. Interesados dirigirse a:

Jordi Gómez
Passeig Marítim, 36
Tel. (977) 23 85 80
43007 Tarragona

Vendo Vic 20 junto con muchos programas, más curso de Basic con libros y cintas. Incluye la cassette. Precio 35.000 ptas.

Daniel Mayans Asian
Apartado 235
Ciudadela (Menorca)
070 Baleares

Vendo juego Decathlon, Dambusters, Baseball por 1.000 ptas. cada uno y Exploding Fist por 1.500 y además cambio juegos para Commodore 64 en cinta.

Pablo Castaño Maldonado
Maestro Guerrero, 3, 7°4
29002 Málaga

Urge vender interface copiador de seguridad. Interface trificador e interface reset.

José Antonio Bogas Mesones
Ramón Patuel, 5, 5°B
Tel. (91) 245 86 68
28017 Madrid

Querría contactar con personas que tuviesen el 128 CMD o si no el 64 CMD, para intercambio de ideas, programas e información, etc... A ser posible de Madrid. Preguntar por:

Daniel Altea Carmona
Cuenca, 41, 2°B
Tel. 699 51 21
Parla (Madrid)

Vendo Commodore 64 con Datassette por 45.000 ptas. y sólo Commodore por 40.000 ptas. Preguntar por:

Andrés
Tel. 656 09 39
Torrejón de Ardoz (Madrid)

Tengo un C-64 y me compre el juego Skyfox pero con las instrucciones en inglés, las necesito en castellano. Pagaré fotocopias. Gracias.

Eladio Ruiz de Lucas
Madre de Dios, 10, 6°C
Alcalá de Henares (Madrid)

Vendo Commodore 64 con gran cantidad de programas y utilidades. También poseo libros y revistas interesantes precio 40.000 ptas. Recién verificado y ajustado. Comprado en Diciembre del 84.

Juan Ignacio Gili Puig
Juan de Mena 5 y 7 13° 2°
Tel. (93) 220 99 66
Barcelona

ERBE

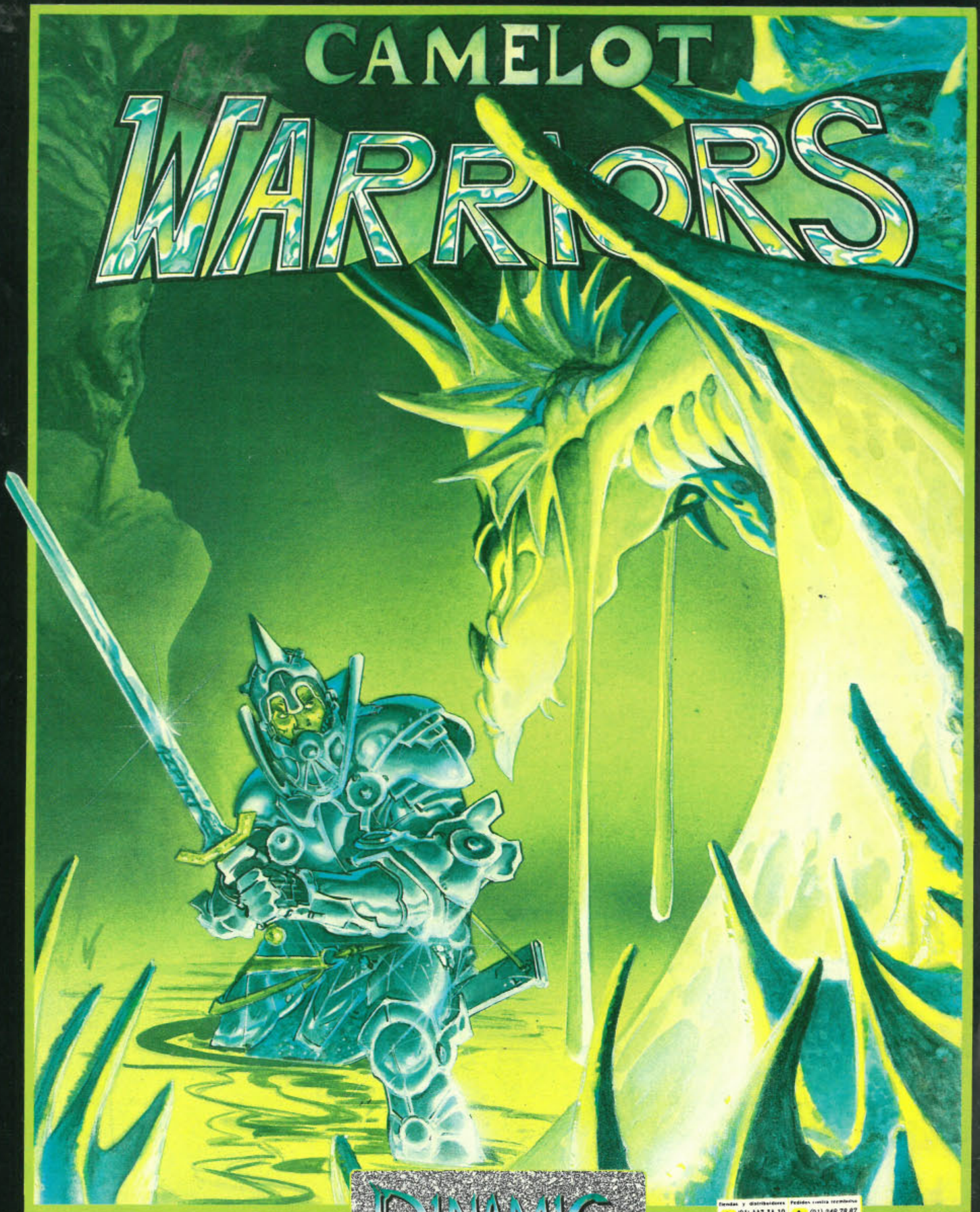
FIST II

LA LEYENDA CONTINUA

La deseada continuación de "EXPLODING FIST"
ya es una realidad; FIST II es mucho más que una segunda parte:
más de 100 pantallas y 700 sprites diferentes en las que
los oponentes al "GRAN MAESTRO" se esconden en bosques, acantilados,
cavernas y lugares con paisajes que te asombrarán.
"EXploding Fist" fue la revelación; **"FIST II", la consagración.**

DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA: ERBE SOFTWARE. C/. STA. ENGRACIA, 17
28010 MADRID. TEL. (91) 447 34 10 - DELEGACION BARCELONA, AVDA. MISTRAL, N.º 10 - TEL. (93) 432 07 31

SPECTRUM · AMSTRAD · COMMODORE · MSX



DINAMIC

Tiendas y distribuidores (91) 447 34 10 Pedidos contra reembolso (91) 248 78 87

DINAMIC SOFTWARE, Plaza de España, 18 · TORRE DE MADRID, 29-1 · 28008 MADRID. TELEX 47008 TRNX-E